(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－259146
（P2002－259146A）
(43)公開日　平成14年９月13日(2002.9.13)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　9/46	３４０	Ｃ０６Ｆ　9/46	３４０Ｆ	５Ｂ０６０
12/00	５９１	12/00	５９１	５Ｂ０９８

審査請求　未請求　請求項の数50　ＯＬ　（全 30 頁）

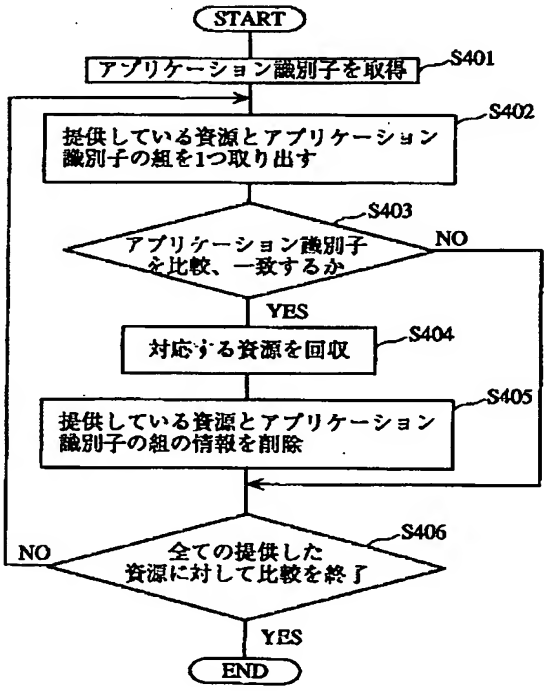
(21)出願番号　特願2001－145592(P2001－145592)
(22)出願日　平成13年５月15日(2001.5.15)
(31)優先権主張番号　特願2000－141492(P2000－141492)
(32)優先日　平成12年５月15日(2000.5.15)
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願2000－141493(P2000－141493)
(32)優先日　平成12年５月15日(2000.5.15)
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願2000－398746(P2000－398746)
(32)優先日　平成12年12月27日(2000.12.27)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者　塩見　隆一
大阪府門真市大字門真1006番地　松下電器産業　株式会社内
(72)発明者　葉山　悟
大阪府門真市大字門真1006番地　松下電器産業　株式会社内
(74)代理人　100090446
弁理士　中島　司朗

最終頁に続く

(54)【発明の名称】　アプリケーション実行装置及び方法

(57)【要約】
【課題】　資源回収をカーネルによらずに実行するアプリケーション実行装置を提供すること、Javaミドルウェアを再起動することなく、連続的にJavaアプリケーションを実行することができるJavaアプリケーション実行装置を提供すること、及びガベージコレクション処理の負荷を軽減したJavaアプリケーション実行装置を提供する。
【解決手段】　ライブラリ部はアプリケーションからの資源の提供の要求に応じて、資源を提供したときに、資源の提供を要求したアプリケーションを識別する識別子を取得し、提供した資源と取得したアプリケーション識別子を組にしてテーブルに保持する。そして、アプリケーションが終了すると、終了したアプリケーションのアプリケーション識別子を取得し、取得した識別子に対応する資源をテーブルから特定し、特定した資源を解放する。



【特許請求の範囲】
【請求項１】　アプリケーションに資源を提供する少なくとも１つ以上のライブラリ部とカーネル部を備えるアプリケーション実行装置であって、
前記カーネル部は、
アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知する通知手段を有し、
各ライブラリ部は、
通知手段から通知されると、終了したアプリケーションに提供した資源を回収する回収手段を有することを特徴とするアプリケーション実行装置。
【請求項２】　前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項1記載のアプリケーション実行装置。
【請求項３】　前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
資源を提供したアプリケーションがどれであるかの通知を前記カーネル部から受け取り、前記テーブルに、資源を提供したアプリケーションと提供した資源を対応付けて組にして登録する登録手段と、
登録した資源が前記回収手段により回収されると、回収された資源と対応するアプリケーションとの対応関係を示す組を前記テーブルから削除する削除手段とを有することを特徴とする請求項２記載のアプリケーション実行装置。
【請求項４】　前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知を行うよう依頼する依頼手段とを有することを特徴とする請求項1記載のアプリケーション実行装置。
【請求項５】　前記依頼手段は、コールバック関数の呼び出しを前記カーネル部に依頼し、
前記カーネル部は、アプリケーションが終了すると、依頼されたコールバック関数を呼び出して実行することにより、前記通知手段として前記通知を行うことを特徴とする請求項４記載のアプリケーション実行装置。
【請求項６】　前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を有し、
前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知を行うよう当該アプリケーションに対応する通知部に依頼する依頼手段とを有することを特徴とする請求項1記載のアプリケーション実行装置。
【請求項７】　前記依頼手段は、前記本ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、
依頼された前記通知部は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行うことを特徴とする請求項６記載のアプリケーション実行装置。
【請求項８】　前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有し、
前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供したライブラリ部に通知し、
前記資源特定手段は、通知された複数のアプリケーションに対応する複数の資源を前記テーブル保持手段により特定し、
前記回収手段は、特定された複数の資源を回収することを特徴とする請求項1記載のアプリケーション実行装置。
【請求項９】　前記各ライブラリ部は、前記資源として、チューナー、MPEGデコーダ、リモコン、ファイルシステム、メモリ、モデムの何れかをアプリケーションに提供することを特徴とする請求項1記載のアプリケーション実行装置。
【請求項１０】　前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項９記載のアプリケーション実行装置。
【請求項１１】　前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する提供手段と、
資源を提供したアプリケーションがどれであるかの通知を前記カーネル部から受け取り、前記テーブルに、資源を提供したアプリケーションと提供した資源を対応付けて登録する登録手段と、
登録した資源が前記回収手段により回収されると、回収

された資源と対応するアプリケーションを前記テーブルから削除する削除手段とを有することを特徴とする請求項１０記載のアプリケーション実行装置。

【請求項１２】　前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知を行うよう依頼する依頼手段とを有することを特徴とする請求項９記載のアプリケーション実行装置。

【請求項１３】　前記依頼手段は、コールバック関数の呼び出しを前記カーネル部に依頼し、
前記カーネル部は、アプリケーションが終了すると、依頼されたコールバック関数を呼び出して実行することにより、前記通知手段として前記通知を行うことを特徴とする請求項１２記載のアプリケーション実行装置。

【請求項１４】　前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を備え、
各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知を行うよう当該アプリケーションに対応する通知部に依頼する依頼手段とを有することを特徴とする請求項９記載のアプリケーション実行装置。

【請求項１５】　前記依頼手段は、前記本ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、
依頼された前記通知部は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行うことを特徴とする請求項１４記載のアプリケーション実行装置。

【請求項１６】　前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有し、
前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供したライブラリ部に通知し、
前記資源特定手段は、通知された複数のアプリケーションに対応する複数の資源を前記テーブル保持手段により特定し、
前記回収手段は、特定された複数の資源を回収することを特徴とする請求項９記載のアプリケーション実行装置。

【請求項１７】　前記通知手段は、さらにアプリケーションが終了した時と中断した時の何れかの時に、前記各ライブラリ部に通知し、
前記各ライブラリ部はさらに、通知されたアプリケーションが終了したか、中断したかに応じて、当該アプリケーションに提供した資源を回収するか否かを判定する判定手段を有し、
前記回収手段は、さらに資源を回収すると判定された場合に、通知されたアプリケーションに提供した資源を回収することを特徴とする請求項1記載のアプリケーション実行装置。

【請求項１８】　前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を備え、
前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了した時と中断した時の何れかの時に、前記通知をしてくれるように当該アプリケーションに対応する通知部に依頼する依頼手段とを有することを特徴とする請求項１７記載のアプリケーション実行装置。

【請求項１９】　前記依頼手段は、前記本ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、
依頼された前記通知部は、前記当該アプリケーションが終了した時と中断した時の何れかの時に、前記資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行い、
前記判定手段は、前記通知を受け取ると、アプリケーションが終了したか、中断したかについての情報を取得し、前記情報に応じて当該アプリケーションに提供した資源を回収するか否かを判定することを特徴とする請求項１８記載のアプリケーション実行装置。

【請求項２０】　アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置であって、
前記Javaミドルウェア部は、
アプリケーションとアプリケーションに対応するタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、
アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに対応するタスクがどれであるかを通知する通知手段とを有し、
前記OS部は、
アプリケーション毎にアプリケーションを実行するタスクを生成するタスク生成手段と、

前記タスクを構成するアプリケーション用スレッドを複数生成するスレッド生成手段と、
前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を提供し、提供した資源と前記スレッドが属するタスクとの対応関係を示すテーブルを保持する管理手段と、
前記通知手段から通知されたタスクに対応する資源を管理手段より特定し、アプリケーションに提供されていた資源を回収する回収手段とを有することを特徴とするアプリケーション実行装置。
【請求項２１】　前記Javaミドルウェア部はさらに、
デバイスの状態の変化を、実行中のアプリケーションに通知してくれるように依頼する依頼手段と、
前記依頼に応じて、前記変化を検知すると、前記アプリケーションに前記変化があったことの通知をする状態変化通知手段とを有することを特徴とする請求項２０記載のアプリケーション実行装置。
【請求項２２】　前記依頼手段は、前記変化があったことの連絡を待ち受けるリスナーの呼び出しを前記状態変化通知手段に依頼し、
前記状態変化通知手段は、前記変化を検知すると、依頼された前記リスナーの呼び出しを行うことにより、前記通知をすることを特徴とする請求項２１記載のアプリケーション実行装置。
【請求項２３】　前記状態変化通知手段は、前記リスナーの呼び出しを実行する専用スレッドを生成し、当該専用スレッドを実行することにより前記リスナーの呼び出しを行い、
前記Javaミドルウェア部はさらに、前記リスナーと前記専用スレッドと前記アプリケーションとの対応関係を示すテーブルを保持する第２テーブル保持手段と、
前記依頼手段が前記リスナーの呼び出しを前記状態変化通知手段に依頼する毎に、前記テーブルを参照し、前記リスナーと対応関係にあるアプリケーションが前記テーブルに保持されているか否かを判定し、保持されている場合は、当該アプリケーションに対応づけて当該リスナーを前記テーブルに追加するテーブル更新手段と、
前記状態変化通知手段は、前記テーブル更新手段によって当該アプリケーションが保持されていると判定された場合は、前記専用スレッドを新たに生成せず、保持されていないと判定された場合は、前記専用スレッドを新たに生成することを特徴とする請求項２２記載のアプリケーション実行装置。
【請求項２４】　前記専用スレッドは、スレッド間の情報伝達を行うキューに、前記変化が発生したことを示す情報が保持されているか否かを監視し、情報が保持されている場合に対応する前記リスナーの呼び出しを行うことを特徴とする請求項２３記載のアプリケーション実行装置。
【請求項２５】　前記専用スレッドは、スレッド間の情報伝達を行うキューに、前記変化が発生したことを示す情報が保持される前はウエイト状態であり、キューに情報が保持されると実行状態に移行し、対応する前記リスナーの呼び出しを行うことを特徴とする請求項２３記載のアプリケーション実行装置。
【請求項２６】　前記Javaミドルウェア部はさらに、
前記タスクを構成する前記アプリケーション用スレッドとは別に、前記Javaミドルウェア部に必要な資源を確保する資源確保用スレッドを生成する資源確保用スレッド生成手段と、
生成した資源確保用スレッドを実行することにより前記Javaミドルウェア部に必要な資源を確保する資源確保手段とを有し、
前記回収手段は、前記資源確保手段によって確保された前記Javaミドルウェア部に必要な資源を回収することなく、前記通知手段から通知されたタスクに対応する資源を前記管理手段より特定し、アプリケーションに提供されていた資源を回収することを特徴とする請求項２０記載のアプリケーション実行装置。
【請求項２７】　アプリケーションに資源を提供するOS部とミドルウェア部を備えるアプリケーション実行装置であって、
前記ミドルウェア部は、
前記OS部からの要求に応じて資源の提供を要求したアプリケーションがどれであるかを通知し、アプリケーションが終了すると、終了したアプリケーションがどれであるかを通知する通知手段を有し、
前記OS部は、
資源の提供を要求したアプリケーションがどれであるかの通知を要求する要求手段と、
通知されたアプリケーションと資源との対応関係を示すテーブルを保持する資源管理テーブル手段と、
通知手段から終了したアプリケーションがどれであるかが通知されると、前記資源管理テーブル手段により終了したアプリケーションに対応する資源を特定し、特定した資源を回収する資源回収手段とを有することを特徴とするアプリケーション実行装置。
【請求項２８】　前記通知手段は、
資源の提供を要求したアプリケーションをロードしたクラスローダを特定するローダ特定手段と、
アプリケーションをロードしたクラスローダとアプリケーションとの対応関係を示すテーブルを保持するテーブル手段と、
前記ローダ特定手段によって特定されたクラスローダに対応するアプリケーションを前記テーブルから特定するアプリケーション特定手段とを有することを特徴とする請求項２７記載のアプリケーション実行装置。
【請求項２９】　前記ローダ特定手段は、アプリケーションを構成するクラスの呼び出し元情報を格納するスタ

ックを参照することにより、資源の提供を要求したアプリケーションをロードしたクラスローダを特定することを特徴とする請求項２８記載のアプリケーション実行装置。

【請求項３０】　前記OS部はさらに、アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項２９記載のアプリケーション実行装置。

【請求項３１】　前記OS部はさらに、

アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項２８記載のアプリケーション実行装置。

【請求項３２】　前記OS部はさらに、アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項２７記載のアプリケーション実行装置。

【請求項３３】　アプリケーションに資源を提供する複数のライブラリ部とカーネル　部を備えるアプリケーション実行装置に用いられるプログラムを記録したコンピュータ　読み取り可能な記録媒体であって、

前記プログラムは、請求項１、１７の何れか１項に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項３４】　アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置に用いられるプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記プログラムは、請求項２０に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項３５】　アプリケーションに資源を提供するOS部とミドルウェア部を備えるアプリケーション実行装置に用いられるプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記プログラムは、請求項２７に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項３６】　ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するアプリケーション実行装置であって、

アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手段と、

起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割当てる割当て手段と、

アプリケーションが終了する毎に、当該アプリケーションに割当てられた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とするアプリケーション実行装置。

【請求項３７】　前記アプリケーション実行装置はさらに、

アプリケーションに関連するオブジェクトに対して、当該アプリケーションに割当てられた分割ヒープ領域内でオブジェクト領域を獲得するオブジェクト領域獲得手段と、

前記分割ヒープ領域を対象にガベージコレクションを行うガベージコレクション手段とを備えることを特徴とする請求項３６記載のアプリケーション実行装置。

【請求項３８】　前記アプリケーション実行装置はさらに、前記ガベージコレクション手段によって対象となる分割ヒープ領域のガベージコレクションが行われている間、当該分割ヒープ領域を使用するアプリケーションのみ実行を停止させ、他の分割ヒープ領域を使用するアプリケーションは継続して実行させるロック手段を備えることを特徴とする請求項３７記載のアプリケーション実行装置。

【請求項３９】　前記アプリケーション実行装置はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係示すテーブルを保持するテーブル手段を備え、

前記メモリ解放手段は、終了したアプリケーションに対応する分割ヒープ領域を前記テーブル手段から特定して解放することを特徴とする請求項３６記載のアプリケーション実行装置。

【請求項４０】　前記アプリケーション実行装置はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係示すテーブルを保持するテーブル手段を備え、

前記割当て手段は、さらに、アプリケーションが起動される毎に、当該アプリケーションと分割領域との新たな対応関係をテーブルに追加することを特徴とする請求項３６記載のアプリケーション実行装置。

【請求項４１】　ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するアプリケーション実行装置であって、

メモリヒープ領域全体をシステムヒープ領域として割当てるシステムヒープ領域割当て手段と、

システム関連のオブジェクト領域をシステムヒープ領域内に獲得するオブジェクト領域獲得手段と、

アプリケーションが起動される毎に、システムヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手

段と、
起動されたアプリケーションに、獲得された分割ヒープ領域を割当てる割当て手段と、
アプリケーションの終了時に、当該アプリケーションに対して割当てられた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とするアプリケーション実行装置。
【請求項４２】　前記アプリケーション実行装置はさらに、システムヒープ領域内部を対象に、分割ヒープ領域を単位にガベージコレクションを行うガベージコレクション手段を備えることを特徴とする請求項４１記載のアプリケーション実行装置。
【請求項４３】　ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するメモリヒープ管理方法であって、
アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得ステップと、
起動されたアプリケーションに、前記分割ヒープ領域獲得ステップによって獲得された分割ヒープ領域を割当てる割当てステップと、
アプリケーションが終了する毎に、当該アプリケーションに割当てられた分割ヒープ領域を解放するメモリ解放ステップとを含むことを特徴とするメモリヒープ管理方法。
【請求項４４】　前記メモリヒープ管理方法はさらに、アプリケーションに関連するオブジェクトに対して、当該アプリケーションに割当てられた分割ヒープ領域内でオブジェクト領域を獲得するオブジェクト領域獲得ステップと、
オブジェクト領域獲得ステップにおいてオブジェクト領域の獲得に失敗したとき、個々の分割ヒープ領域を対象にガベージコレクションを行うガベージコレクションステップとを含むことを特徴とする請求項４３記載のメモリヒープ管理方法。
【請求項４５】　前記メモリヒープ管理方法はさらに、前記ガベージコレクションステップによって対象となる分割ヒープ領域のガベージコレクションが行われている間、当該分割ヒープ領域を使用するアプリケーションのみ実行を停止させ、他の分割ヒープ領域を使用するアプリケーションは継続して実行させるロックステップを含むことを特徴とする請求項４４記載のメモリヒープ管理方法。
【請求項４６】　前記メモリヒープ管理方法はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係示すテーブルを保持するテーブル保持ステップを含み、
前記メモリ解放ステップは、終了したアプリケーションに対応する分割ヒープ領域を前記テーブルから特定して解放することを特徴とする請求項４３記載のメモリヒープ管理方法。
【請求項４７】　前記メモリヒープ管理方法はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係示すテーブルを保持するテーブル保持ステップを含み、
前記割り当てステップは、さらに、アプリケーションが起動される毎に、当該アプリケーションと分割領域との新たな対応関係を前記テーブルに追加することを特徴とする請求項４３記載のメモリヒープ管理方法。
【請求項４８】　ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するメモリヒープ管理方法であって、
メモリヒープ領域全体をシステムヒープ領域として割当てるシステムヒープ領域割当てステップと、
システム関連のオブジェクト領域をシステムヒープ領域内に獲得するオブジェクト領域獲得ステップと、
アプリケーションが起動される毎に、システムヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得ステップと、
起動されたアプリケーションに、獲得された分割ヒープ領域を割り当てる割当てステップと、
アプリケーションの終了時に、当該アプリケーションに対して割当てられた分割ヒープ領域を解放するメモリ解放ステップとを含むことを特徴とするメモリヒープ管理方法。
【請求項４９】　前記メモリヒープ管理方法はさらに、分割ヒープ領域獲得ステップが分割ヒープ領域の獲得に失敗したとき、システムヒープ領域内部を対象に、分割ヒープ領域を単位にガベージコレクションを行うガベージコレクションステップを含むことを特徴とする請求項４８記載のメモリヒープ管理方法。
【請求項５０】　ガベージコレクションが必要なアプリケーションを実行する装置に用いられるプログラムを記録したコンピュータ読み取り可能な記録媒体であって、
前記プログラムは、請求項４３～４９の何れか1項に記載のステップをコンピュータに実行させることを特徴とする記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、アプリケーション実行装置において、アプリケーション終了により不必要になった資源の回収を実行する技術に関する。
【０００２】
【従来の技術】一般的なアプリケーション実行装置においては、オペレーティングシステムのカーネルが、アプリケーション終了等により不要になった資源（メモリ、キーボードなどの各種デバイス）の回収を行っている。ここで、「回収」とは、カーネル等のオペレーションシステム側がアプリケーションに提供した資源をアプリケ

ーションから取り返すことをいう。カーネルは資源の回収を行うために、資源の回収の手順をデバイス毎に記憶している。これにより、アプリケーションが使用していたデバイスが不要になると、カーネルは当該デバイスの回収の手順に従って、資源を回収することができ、資源を効率よく利用することができる。

【０００３】また、Javaアプリケーション実行装置において、例えばPCでは、メモリ以外の他の資源の回収については、Javaミドルウェア（いわゆるバーチャルマシン）自体は資源の回収を行わず、Javaミドルウェアが終了した時に、カーネルによって資源の回収が行われる。一方、メモリの回収は、Javaミドルウェアによる「ガベージコレクション」により行われる。「ガベージコレクション」とは、メモリ領域中の不必要になった領域を回収することをいう。Javaでは、アプリケーションのロードは、アプリケーションを構成する複数のクラスの各インスタンス（以下オブジェクトと記述）をメモリ中にロードすることにより行われる。メモリ領域は、実行中のアプリケーションのためにロードされるオブジェクトで共有され、メモリ領域には、ロードされる順に領域が割当てられる。従って、複数のアプリケーションから連続的にオブジェクトがメモリ領域にロードされた場合には、１つのアプリケーションからロードされた各オブジェクトがメモリ領域に不連続な形で割当てられることになる。メモリ領域がロードされたオブジェクトで不連続に占有されると、メモリ不足を解消するため、ガベージコレクションが行われる。

【０００４】

【発明が解決しようとする課題】しかしながら、上記に説明した資源回収処理の技術には以下に示す問題点がある。第１に、アプリケーション実行装置におけるカーネルによる資源の回収処理の場合には、新規にデバイスをアプリケーション実行装置に追加する毎に、追加されたデバイスの回収手順を新たにカーネルに記憶させなければならないという問題がある。

【０００５】第２に、Javaアプリケーション実行装置における資源の回収処理の場合には、Javaミドルウェアを終了することなく、アプリケーションを逐次的に実行してゆくと、カーネルによるメモリ以外の資源の回収は、Javaミドルウェアが終了するまで行われないため、１つのアプリケーションが終了しても、そのアプリケーションに提供された資源は回収されないので、次のアプリケーションが同じ資源を必要とする場合には、次のアプリケーションが実行できなくなるという問題がある。

【０００６】第３に、不連続な形でオブジェクトに割当てられたメモリ領域が不要となった場合、メモリ領域のガベージコレクションの負荷が大きくなるという問題がある。上記の問題点に鑑み、本発明は、資源回収をカーネルによらずに実行するアプリケーション実行装置を提供すること、Javaミドルウェアを再起動することなく、連続的にJavaアプリケーションを実行することができるJavaアプリケーション実行装置を提供すること、及びガベージコレクション処理の負荷を軽減したJavaアプリケーション実行装置を提供することを目的とする。

【０００７】

【課題を解決するための手段】上記課題を解決するために、本発明に係るアプリケーション実行装置は、アプリケーションに資源を提供する少なくとも１つ以上のライブラリ部とカーネル部を備えるアプリケーション実行装置であって、前記カーネル部は、アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知する通知手段を有し、各ライブラリ部は、通知手段から通知されると、終了したアプリケーションに提供した資源を回収する回収手段を有することを特徴とする。本アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【０００８】また、本発明に係るアプリケーション実行装置は、アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置であって、前記Javaミドルウェア部は、アプリケーションとアプリケーションに対応するタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに対応するタスクがどれであるかを通知する通知手段とを有し、前記OS部は、アプリケーション毎にアプリケーションを実行するタスクを生成するタスク生成手段と、前記タスクを構成するアプリケーション用スレッドを複数生成するスレッド生成手段と、前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を提供し、提供した資源と前記スレッドが属するタスクとの対応関係を示すテーブルを保持する管理手段と、前記通知手段から通知されたタスクに対応する資源を管理手段より特定し、アプリケーションに提供されていた資源を回収する回収手段とを有することを特徴とする。本アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【０００９】本発明に係るアプリケーション実行装置は、ガベージコレクションが必要なアプリケーションを実行する装置であって、アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手段と、起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割当てる割当て手段と、アプリケーションが終了する毎に、当該アプリケーションに割当てら

れた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とする。本アプリケーション実行装置における各手段等をステップとするメモリヒープ管理方法としたり、それらステップをコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【００１０】

【発明の実施の形態】（実施の形態１）本実施の形態におけるアプリケーション実行装置においては、アプリケーションが終了等する毎にカーネル部ではなく、ライブラリ部がアプリケーションに提供した資源の回収を行う。ここで、「アプリケーション」は、プログラムとプログラムが実行されることによる機能の何れかを指すものとする。「資源」とは、チューナー、MPEGオーディオデコーダ、MPEGビデオデコーダ、キーボード入力デバイス、リモコン入力デバイス、マウス入力デバイス、トラックボール入力デバイス、ミューテックス、セマフォア、ネットワークデバイス、シリアルデバイス、IEEE1394（シリアルインターフェース）、USB（Universal Serial Bus）デバイスインターフェース、ファイルシステム、ディスプレイデバイス、メモリ、モデム等をいう。ライブラリ部は、アプリケーションの要求に応じて資源を制御（資源の確保、管理、回収）するいわゆるデバイスドライバ群である。「回収」とは、カーネル等のオペレーションシステム側がアプリケーションに提供した資源をアプリケーションから取り返すことをいう。

【００１１】具体的には、ライブラリ部はアプリケーションからの資源の提供の要求に応じて、資源を提供したときに、資源の提供を要求したアプリケーションを識別する識別子を取得し、提供した資源と取得したアプリケーション識別子を組にしてテーブルに保持する。そして、アプリケーションが終了すると、終了したアプリケーションのアプリケーション識別子を取得し、取得した識別子に対応する資源をテーブルから特定し、特定した資源を回収する。

【００１２】以下本アプリケーション実行装置の実施の形態について、図面を用いて説明する。図1は、アプリケーション実行装置１０の構成を示すブロック図である。アプリケーション実行装置１０は、アプリケーション入力部１１、アプリケーション記憶部１２、カーネル部１３、ライブラリ格納部１４、デバイス部１５、指示通知部１６から構成される。

【００１３】アプリケーション入力部１１は、フロッピー（登録商標）ディスクドライブ、CDドライブ、ネットワークインターフェースボード、放送受信機器等で構成され、実行すべきアプリケーションを受け付け、アプリケーション記憶部１２に記憶させる。アプリケーション記憶部１２は、RAM、ROM、ハードディスク、CDドライブ、フロッピーディスク等で構成され、アプリケーション入力部１１から出力されたアプリケーション又は予め内蔵されているアプリケーションを記憶する。

【００１４】カーネル部１３は、逐次実行部１３a、資源回収通知部１３b、識別子提供部１３cから構成され、アプリケーション記憶部１２が記憶するアプリケーションを実行する。逐次実行部１３aは、実行するアプリケーションがバイナリープログラムの場合、バイナリープログラムをそのまま実行するCPU等で構成される。実行するアプリケーションがJavaバイトコード(Javaは、米国のSun Microsystems, Inc.の商標である。)のような中間コードの場合は中間コードを逐次解析し、実行するモジュール（バーチャルマシンやインタプリタ）等で構成される。逐次実行部１３aは、アプリケーション記憶部１２が記憶するアプリケーションを読み出し、実行する。さらに、実行するアプリケーションからライブラリ格納部１４に格納されるライブラリ部に資源の提供を要求し、ライブラリ部からアプリケーションが要求する資源の提供を受ける。

【００１５】資源回収通知部１３bは、アプリケーションが終了等する毎に当該アプリケーションに資源を提供しているライブラリ部に対して、資源回収の指示と当該アプリケーション識別子とを通知する。具体的には、資源回収通知部１３bは、ライブラリ格納部１４が格納するライブラリ部から、コールバック関数の登録を受けつけ、指示通知部１６からアプリケーションの終了又は中断の通知を受けると、登録されているコールバック関数を呼び出して実行し、当該アプリケーションの識別子をライブラリ部に通知する。

【００１６】ここで、「コールバック関数の登録」とは、ライブラリ部がアプリケーションの要求に応じて、当該アプリケーションに新たな資源の提供を開始したときに、「当該ライブラリ部に、当該アプリケーションが終了又は中断すれば中断又は終了したアプリケーションがどれであるかを、コールバック関数を呼び出して実行することにより通知してくれるように」資源回収通知部１３bに依頼することをいう。

【００１７】図2は、コールバック関数を定義したC言語プログラムの一例を示す。CALLBACK#fは、コールバック関数の型名を示す。このコールバック関数は返り値がなく、引数に整数の値をとる。図3は、コールバック関数の登録と削除と実行を記述したC言語プログラムの例を示す。行番号6～14は登録のプログラム例を示し、行番号16-20は削除のプログラム例を示し、行番号22～26は実行のプログラム例を示す。

【００１８】識別子提供部１３cは、ライブラリ部からの要求に応じて、アプリケーションの識別子を提供する。ライブラリ格納部１４は、資源であるデバイスを制御する複数のライブラリ部１４a～１４ｎからなる。各ライブラリ部は、デバイス部１５の対応するデバイスを制御する（例えばライブラリ部１４aは、デバイス１５aを、ライブラリ部１４bは、デバイス１５bを制御す

る。）。実際にはライブラリ格納部１４は、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の関数群を記述したプログラムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

【００１９】具体的には、各ライブラリ部は、以下に示す処理を行うことにより資源を制御する。アプリケーションの要求に応じて資源を提供し、資源回収通知部１３bにコールバック関数を登録する。次にアプリケーション識別部１３cから資源を要求したアプリケーションのアプリケーション識別子を取得し、提供した資源の資源名と組にしてテーブルに保持する。さらに、登録したコールバック関数が資源解放通知部１３bにより実行され、回収すべき資源を使用しているアプリケーションのアプリケーション識別子が通知されると、保持しているテーブルから通知されたアプリケーション識別子に対応する資源名を特定し、特定した資源名の資源を回収する。

【００２０】次に各ライブラリ部がアプリケーションの要求に応じて行う資源提供の処理について図４を参照して説明する。図４は、各ライブラリ部が行う資源提供の処理を示すフローチャートである。各ライブラリ部は、アプリケーションの要求に応じて、資源を確保し（S301）、コールバック関数を既に登録しているか否かを判断（例えばフラグを設定し、登録・未登録を判断）し（S302）、登録していない場合（S302:N）は、資源回収通知部１３bにコールバック関数を登録する（S303）。

【００２１】次に識別子提供部１３cを呼び出し、資源を要求したアプリケーションのアプリケーション識別子を取得し（S304）、取得したアプリケーション識別子と確保した資源の資源名を組にしてテーブルに保持し（S305）、資源をアプリケーションに提供する（S306）。登録している場合（S302:Y）は、コールバック関数を新たに登録することなく、S304～S306の処理を実行する。

【００２２】上記テーブルの例を図５に示す。図５は、ファイルシステムを制御するライブラリ部（以下ライブラリ部Fと呼ぶ。）が保持するアプリケーション識別子と資源名（ファイル名）とを組にしたテーブルを示す。図５－（１）は、ライブラリ部Fが資源解放通知部１３bにコールバック関数を登録し、アプリケーション識別子「１」のアプリケーションに対し、ファイル名「a.txt」の資源を提供していることを示す。アプリケーションがライブラリ部Fの関数を呼び出すことにより、資源の提供を要求すると、ライブラリ部Fは、ファイル名「b.txt」の資源を確保し、識別子提供部１３cからアプリケーション識別子「２」を取得する。この場合、ライブラリ部Fのコールバック関数は、既に登録済みなので再登録はされない。ライブラリ部Fは、取得したアプリケーション識別子「２」と確保した資源のファイル名「b.txt」を組としてテーブルに保持し、図５－（２）に示すようにテーブルを更新する。

【００２３】次に各ライブラリ部が行う資源の回収処理について図６を参照して説明する。図６は、各ライブラリ部が行う資源の回収処理を示すフローチャートである。コールバック関数が呼び出されると、各ライブラリ部はアプリケーション識別子を資源回収通知部１３bより取得し（S401）、テーブルに保持するアプリケーション識別子と資源名の組を１つ取り出し（S402）、取り出したアプリケーション識別子と資源回収通知部１３bより取得したアプリケーション識別子を比較する（S403）。アプリケーション識別子が一致する場合は、取り出した組の資源名の資源を回収し（S404）、テーブルから当該組を削除し（S405）、テーブルに保持する全ての組との比較が終了している場合（S406：Y）は処理を終了する。例えば図５－（２）のテーブルの例で、取得したアプリケーション識別子が「２」の場合には、テーブルからアプリケーション識別子「２」と対応するファイル名「b.txt」の組が削除される（図５－（３））。全ての組との比較が終了していない場合（S406:N）は次の組を取り出し（S402）、同様の処理を繰り返す（S403～S406）。アプリケーション識別子が一致しない場合は、テーブルに保持する全ての組との比較が終了している場合（S406：Y）は処理を終了する。全ての組との比較が終了していない場合（S406:N）は次の組を取り出し（S402）、同様の処理を繰り返す（S403～S406）。

【００２４】デバイス１５a～１５nは、メモリー ディスプレイ、入力デバイス（キーボード、リモコン等）、ファイルシステム、ネットワークデバイスなどを含み、各デバイスはライブラリ格納部１４が格納する各ライブラリ部を介してなされるアプリケーションからの制御に応じて作動する。指示通知部１６は、アプリケーションの開始、停止、再開、終了の処理をする。

【００２５】アプリケーションの開始の処理とは、逐次実行部１３aにアプリケーション記憶部１２が記憶するアプリケーションの実行を指示することをいう。アプリケーション停止の処理とは、逐次実行部１３aに実行しているアプリケーションを中断させ、必要に応じて資源回収通知部１３bにアプリケーションの中断を通知することをいう。

【００２６】アプリケーション再開の処理とは、逐次実行部１３aに中断しているアプリケーションの実行を指示することをいう。アプリケーション終了の処理とは、逐次実行部１３aに実行しているアプリケーションの終了を指示し、資源回収通知部１３bにアプリケーションの終了を通知することをいう。

【００２７】本実施形態においては、アプリケーション終了又は中断時に資源をアプリケーションに提供した各ライブラリ部に通知されるアプリケーション識別子は、1つであったが、複数のアプリケーションが同時に終了する場合は、複数のアプリケーション識別子を、各ライ

ブラリ部に通知することとしてもよい。この場合の各ライブラリ部が行う資源の回収処理のフローチャートを図７に示す。各ライブラリ部は回収する資源を提供した複数のアプリケーション識別子を資源回収通知部１３ｂより取得し（S751）、取得したアプリケーション識別子を１つ取り出し（S752）、テーブルに保持するアプリケーション識別子と資源名の組を１つ取り出し（S753）、取り出したアプリケーション識別子と資源回収通知部１３ｂより取得したアプリケーション識別子を比較する（S754）。アプリケーション識別子が一致する場合（S754:Y）は、対応する資源名を回収対象の資源名として保持し（S755）、テーブルから当該組を削除し（S756）、テーブルに保持する全ての組との比較が終了している場合（S757：Y）は、次のアプリケーション識別子を取得し（S758、S752）、同様の処理を繰り返す。S751で取得した全てのアプリケーション識別子に対して比較が終了すると（S758：Y）、S755で回収対象の資源名として保持している資源名に対応する資源を一括して回収する（S759）。

【００２８】これにより、ライブラリ部は、効果的に資源回収を行うことができる。例えば、（１つの）ライブラリ部が管理する資源が、複数のアプリケーションの終了によって全て回収される場合、管理テーブルを初期化する処理だけで迅速に資源を回収することができる。

（実施の形態２）実施形態１においては、ライブラリ部は、アプリケーションが終了等した場合に呼び出されるコールバック関数をライブラリ部毎に登録したが、本実施形態においては、実施形態１のコールバック関数に相当する資源回収インスタンスを生成し、アプリケーションが終了等する毎にアプリケーションに提供した資源を回収する。ここで、「資源回収インスタンス」は、ライブラリ部が格納するクラス（Javaバイトコードのプログラム）から生成され、各種の情報とその情報を取り出すための関数であるメソッドが記述されたプログラムであり、資源回収インスタンスは、資源の提供を要求したアプリケーションと提供した資源の情報とこれらの情報を取り出すためのメソッドを保持している。このメソッドをプログラム上で呼び出すことにより保持している情報が取り出される。

【００２９】以下本アプリケーション実行装置の実施の形態について、図面を用いて説明する。図８は、アプリケーション実行装置２０の構成を示すブロック図である。アプリケーション実行装置２０は、アプリケーション入力部１１、アプリケーション記憶部１２、OS部２３、ライブラリ格納部２４、デバイス部１５、指示通知部１６から構成される。アプリケーション入力部１１、アプリケーション記憶部１２、デバイス部１５、指示通知部１６は、実施形態１の実行装置１０に含まれるものと同一であるので、説明を省略する。

【００３０】OS部２３は、バーチャルマシン部２３a、アプリケーション管理部２３bから構成され、アプリケーション記憶部１２が記憶するJavaアプリケーション（以下アプリケーションという。）を実行する。バーチャルマシン部２３aは、アプリケーション記憶部１２からアプリケーションをロードし、アプリケーションのバイトコードを逐次解析し、アプリケーションを実行する。また、ライブラリ格納部２４に格納される各ライブラリ部を介してデバイス部１５に格納される各デバイスを制御する。実際には、バーチャルマシン部２３aが実行するアプリケーションからデバイス制御を記述したプログラムである各ライブラリ部のクラスが呼び出され、クラスのプログラムが実行されることにより、各デバイスの制御が実現される。

【００３１】アプリケーション管理部２３bは、実行されているアプリケーションを管理し、また、実施形態１の識別子提供部１３cに相当する機能を有する。具体的には、アプリケーション管理部２３bは、アプリケーション情報を格納する処理を定義するクラスから、処理を実行するインスタンスをアプリケーション毎に生成し、生成した各インスタンス（ここでは、アプリケーション情報インスタンスと呼ぶ。）に、実行されている各アプリケーションのアプリケーション識別子を格納させ、生成した各アプリケーション情報インスタンス（図８の２３c～２３e）を用いて実行されているアプリケーションを管理する。また各ライブラリ部がアプリケーションから資源の提供を要求されたとき、要求したアプリケーションを特定する識別子としてアプリケーション情報インスタンスを提供する。ここで、各アプリケーションの識別子を各アプリケーション情報インスタンスに格納させる代わりに、アプリケーション情報インスタンスそのものをアプリケーションの識別子としてもよい。

【００３２】各アプリケーション情報インスタンスは、実施形態１の資源回収通知部１３bに相当する機能を有する。具体的には、以下の処理を行う。各アプリケーション情報インスタンスは、各ライブラリ部から資源回収インスタンスの登録を受け付ける。ここで、「資源回収インスタンスの登録」とは、ライブラリ部がアプリケーションに新たな資源を提供したときに、「当該ライブラリ部に、当該アプリケーションが終了又は中断すれば、当該アプリケーションがどれであるかを通知してくれるように」当該アプリケーション識別子を格納するアプリケーション情報インスタンスに依頼することをいう。上記の通知は、登録時に生成される資源回収インスタンスに実装されるメソッドを呼び出すことにより実行される。資源回収インスタンスは各ライブラリ部が格納する各クラスから、以下に示すように生成される。各クラスは、「インターフェース」を実装することにより、「インターフェース」が定義するメソッドを実装する。「インターフェース」は、クラスに実装されるべきメソッドを定義する。図９は、このインターフェースの一例であ

る。この例では、ResourceCollectionListenerインターフェースがupdateメソッドを定義している。各ライブラリ部が格納する各クラスは、このインターフェースを実装することにより、updateメソッドに資源回収処理を記述したクラスを用意し、用意したクラスから当該インターフェースを実装した資源回収インスタンスが生成され、アプリケーション情報インスタンスに生成した資源回収インスタンスが登録される。この処理により、生成された資源回収インスタンスには、インターフェースが定義するメソッド（ここでは、updateメソッド）が実装されていることが保証され、上記の通知はこのインターフェースが定義するメソッドを呼び出すことにより実行される。

【００３３】また、アプリケーションが終了又は中断した場合に、当該アプリケーション識別子を格納するアプリケーション情報インスタンスは、登録された資源回収インスタンスに実装されているメソッド（ここでは、updateメソッド）を呼び出すことにより、当該アプリケーションに資源を提供しているライブラリ部に上記の通知をする。

【００３４】さらに、アプリケーション情報インスタンスは、登録された資源回収インスタンスを削除するメソッドを格納してもよいし、アプリケーションの状態を知るためのメソッドを格納してもよい。図１０は、上記のメソッドを格納するアプリケーション情報インスタンスを生成するクラスの一例を示す。図１０において、アプリケーション情報インスタンスを生成するクラスであるapplicationPropoxyは、アプリケーション名(application name)に相当する内部変数と、アプリケーションの状態を知るためのgetStatusメソッド、資源回収インスタンス（resourceCollectionLister）を登録するaddListenerメソッド、資源回収インスタンスの登録を削除するremoveListenerメソッドを定義している。

【００３５】ライブラリ格納部２４は、複数のライブラリ部２４a～２４ｎからなる。各ライブラリ部は、デバイス部１５のデバイス１５a～１５ｎを制御し、アプリケーションからの要求に応じて資源であるデバイス提供の処理と提供した資源回収の処理を行う。実際にはライブラリ格納部２４は、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用のプログラムであるクラス群に記述されたプログラムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

【００３６】最初に資源提供の処理について説明する。各ライブラリ部は、アプリケーションから資源の提供を要求されると、アプリケーションに要求された資源を提供し、アプリケーション管理部２３bを呼び出し、当該アプリケーションに対応するアプリケーション情報インスタンスを取得し、資源回収クラスから資源回収インスタンスを生成し、生成した資源回収インスタンスを当該アプリケーション情報インスタンスに登録する。さらに、各ライブラリ部は、提供した資源の資源名と当該アプリケーション情報インスタンスとを組にして資源回収インスタンスに保持する。

【００３７】図１１に資源回収クラスの一例を示す。図１１においては、ファイルシステムを制御する資源回収クラスが定義されている。具体的には、資源回収インスタンス生成時に作成される「フィールド」と呼ばれる内部変数app（アプリケーション情報インスタンスに相当する）とname（資源のファイル名に相当する）の情報とアプリケーション終了時又は中断時に呼び出され、資源回収の処理を実行するupdateメソッドが定義されている。図１２は、図１１の資源回収クラスが生成した資源回収インスタンスの例を示す。９０１は資源回収インスタンスを保持するインスタンステーブルを示し、９０２、９０３は、生成された資源回収インスタンスを示す。図１２－（１）は、資源回収情報インスタンス９０２が提供した資源名（ここでは、ファイル名「a.txt」）とアプリケーション情報インスタンス（ここでは、「１」とする。）を組にして保持していることを示す。図１２－（２）は、ライブラリが図１２－（１）とは別のアプリケーション情報インスタンス「２」に管理されるアプリケーションに資源（ファイル名「b.txt」のファイル）を提供し、新たに資源回収インスタンス９０３を生成したことを示す。

【００３８】以下図１３を用いて、資源提供の処理をさらに詳しく説明する。図１３は、各ライブラリ部が行う資源提供の処理を示すフローチャートである。各ライブラリ部は、アプリケーションから資源の提供を要求されると、要求に応じて資源を確保し（S1001）、アプリケーション管理部２３ｂから当該アプリケーションに対応するアプリケーション情報インスタンスを取得する（S1002）。次に保持している資源回収インスタンスの１つを取り出し（S1003）、アプリケーション管理部２３bから取得したアプリケーション情報インスタンスと取り出された資源回収インスタンスに保持されるアプリケーション情報インスタンスを比較する（S1004）。アプリケーション情報インスタンスが一致する場合（S1004:Y）は、当該アプリケーション情報インスタンスを保持する資源回収インスタンスに確保した資源の資源名を保持し（S1005）、確保した資源を当該アプリケーションに提供する（S1009）。一方、S1004において、アプリケーション情報インスタンスが一致しない場合（S1004:N）には、保持している他の全ての資源回収インスタンスに対してアプリケーション情報インスタンスの比較を行い（S1006）、一致するものがなければ（S1006:Y）、新たに資源回収インスタンスを生成し、生成した資源回収インスタンスに、アプリケーション管理部２３bから取得したアプリケーション情報インスタンスと確保した資源の資源名を組にして保持し（S1007）、生成した資源回

収インスタンスを資源の確保を要求したアプリケーションに対応するアプリケーション情報インスタンスに登録し（S1008）、確保した資源をアプリケーションに提供する（S1009）。

【００３９】次に資源回収の処理について説明する。ライブラリ部は、アプリケーション終了又は中断時に当該アプリケーション識別子を格納するアプリケーション情報インスタンスから対応する資源回収インスタンスのメソッドが呼び出されると、当該資源回収インスタンスに保持されている資源名に対応する資源を回収する。上記の場合において、アプリケーション情報インスタンスにアプリケーションの状態を知るためのメソッド（以下アプリケーション状態メソッドという。）と資源回収インスタンスの登録を削除するメソッドとを用意した場合の資源回収の処理について説明する。図１４は、アプリケーション情報インスタンスがアプリケーション状態メソッドを有している場合のライブラリ部が行う回収の処理を示すフローチャートである。ライブラリ部は、アプリケーション情報インスタンスから登録した資源回収インスタンスのメソッドが呼び出されると、資源回収インスタンスに保持されているアプリケーション情報インスタンスのアプリケーション状態メソッドを呼び出し、アプリケーションが中断されたのか、終了されたかの情報を取得し（S1101）、取得した情報に基いて資源を回収するか否かを判定し（S1102）、資源を回収する場合は、当該資源回収インスタンスに保持されている資源名に対応する資源を回収し（S1103）、当該資源回収インスタンスの登録を削除するメソッドを呼び出し、当該メソッドを実行することにより、当該資源回収インスタンスを解放する（S1104）。

【００４０】なお、S1102における判定は、アプリケーションが使用している全ての資源ではなく、特定の一部の資源を回収するという判定であってもよい。この場合、S1104における資源回収インスタンスの解放は、全ての資源が回収されたときに行われる。

（実施の形態３）本実施の形態におけるアプリケーション実行装置は、アプリケーション毎にスレッドを生成し、生成したスレッドでアプリケーションが要求する資源を確保し、アプリケーションが終了する毎に終了したアプリケーションに提供した資源の解放を、生成したスレッドと対応付けられたタスク単位でOS部が行うことにより、不要な資源を回収し、Javaミドルウェア（いわゆるバーチャルマシン）を終了することなく次のアプリケーションの実行を可能とするものである。ここで、「スレッド」とは、アプリケーションのプログラムコードを逐次実行するコンテキスト（一連の処理の流れ）である。Javaで記述されたアプリケーションは、複数のスレッドを生成し、それぞれのスレッドがアプリケーションのプログラムコードを並行して実行させることができる。本実施形態において、「タスク」とは、アプリケーションを実行する複数のスレッドの集合である。本実施形態においてアプリケーションに割当てられる資源は、この「タスク」と関連づけて管理される。

【００４１】以下、本アプリケーション実行装置の実施の形態について図面を用いて詳細に説明する。図１５は、アプリケーション実行装置３０の構成を示すブロック図である。アプリケーション実行装置３０は、アプリケーション入力部１１、アプリケーション記憶部１２、Javaミドルウェア部３３、OS部３４、ハードウェア部３５、指示通知部３６で構成される。アプリケーション入力部１１、アプリケーション記憶部１２は、実施形態１の実行装置１０に含まれるものと同一であるので、説明を省略する。

【００４２】Javaミドルウェア部３３は、VM（バーチャルマシン）部３３a、アプリケーション管理部３３b、クラスライブラリ格納部３３cから構成され、アプリケーション記憶部１２が記憶するアプリケーションを実行する。VM部３３aは、アプリケーションを実行する。具体的には、VM部３３aは、以下に示す処理を行う。バイトコードで記述されたアプリケーションをハードウェア部３５に含まれるCPU３５aが実行できるバイナリコードに逐次翻訳し、翻訳したバイナリコードをCPU３５aに実行させる。

【００４３】また、VM部３３aは、自己の動作とアプリケーションの実行に必要な資源を確保する。具体的には、VM部３３aは、クラスライブラリ格納部３３cに含まれる各クラスライブラリ部を呼び出し、各クラスライブラリ部を介して自己の動作に必要な資源を確保する。また、アプリケーション実行中にアプリケーションから資源の確保を要求されると、クラスライブラリ格納部３３cに含まれる各クラスライブラリ部を呼び出し、各クラスライブラリ部を介してアプリケーションが要求した資源を確保する。

【００４４】また、VM部３３aは、アプリケーションから処理の終了したスレッドの削除を要求されると、カーネル３４aに依頼してスレッドを削除させる。アプリケーション管理部３３bは、アプリケーション起動の処理、アプリケーション実行中に追加又は削除されるスレッドの管理、アプリケーション終了の処理を行う。

【００４５】最初に、アプリケーション起動の処理について説明する。アプリケーション管理部３３bは、指示通知部３６からアプリケーションの開始指示を受けると、カーネル３４aにタスクと、タスクに属する最初のスレッドを生成させ、VM部３３aに生成した最初のスレッドで、アプリケーションを実行させることにより、VM部３３aの動作に必要な資源を確保させ、アプリケーションを起動させる。

【００４６】次にアプリケーション実行中に追加又は削除されるスレッドの管理について説明する。アプリケーション管理部３３bは、アプリケー

ションに対応して生成されたタスクとタスクを構成するスレッドとの対応関係を示すテーブルを保持し、アプリケーションの要求によりスレッドが追加又は削除される毎にテーブルを更新することにより、スレッドをアプリケーションとタスクに対応づけて管理する。図16にテーブルの例を示す。図16－（1）では2つのアプリケーションが実行中であり、アプリケーションID「1」のアプリケーションに対し、タスクID「201」のタスクが生成され、タスクID「201」のタスクはスレッドID「1」と「2」のスレッドで構成され、アプリケーションID「2」のアプリケーションに対し、タスクID「202」のタスクが生成され、タスクID「202」のタスクはスレッドID「4」、「5」、「6」のスレッドにより構成されていることを示し、図16－（2）は、アプリケーションID「1」のアプリケーションを実行中に、新たなスレッド（スレッドID「7」のスレッド）が生成され、生成されたスレッドのスレッドID「7」がテーブルに追加されたことを示し、図16－（3）は、アプリケーション「2」のアプリケーションを実行中にスレッドID「5」のスレッドが削除されたことを示す。

【0047】最後に、アプリケーション終了の処理について説明する。図17は、アプリケーション終了時にアプリケーション管理部33bが行う処理を示すフローチャートである。アプリケーション管理部33bは、指示通知部36からアプリケーション終了の指示を受けると、カーネル34aに終了したアプリケーションに対応するタスクのIDを通知し、資源回収を依頼し（S801）、次に、クラスライブラリ格納部33cに含まれる各クラスライブラリ部に終了するアプリケーションのIDを通知し（S802）、最後に終了したアプリケーションに対応するタスクを生成するために確保したメモリを解放してタスクを終了させる（S803）ことにより、アプリケーション終了の処理を行う。例えば、図16－（3）の例では、アプリケーションID「2」のアプリケーションが終了した場合、アプリケーション管理部33bは、カーネル34aにアプリケーションID「2」に対応するタスクID「202」を通知し、カーネル34aにタスクID「202」に対応する資源の回収を依頼し、次にクラスライブラリ部に終了したアプリケーションID「2」を通知し、最後にメモリを解放して生成したタスク「202」及びその中に含まれるスレッド「4」と「6」を終了させる。

【0048】クラスライブラリ格納部33cは、複数のクラスライブラリ部33c1～33cnからなり、各クラスライブラリ部は、ハードウェア部35に含まれる資源であるデバイスを制御し、資源提供の処理、リスナーの管理とリスナーの呼び出しの処理を行う。実際には、クラスライブラリ格納部33cは、ROMやハードディスクなどで構成され、各クラスライブラリ部では、デバイス制御用のプログラムであるクラス群に記述されたプログラムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

【0049】最初に、各クラスライブラリ部が行う資源提供の処理について説明する。クラスライブラリ部は、VM部33aが実行するアプリケーションから呼び出され、資源であるデバイスの提供を要求されると、OS部34のライブラリ格納部34bに格納される当該デバイスに対応するライブラリ部を呼び出し、要求された資源を提供する。

【0050】次に各クラスライブラリ部が行うリスナー管理の処理について説明する。クラスライブラリ部は、VM部33aが実行するアプリケーションから呼び出され、リスナー登録され、登録されたリスナーを呼び出す処理をするスレッド（以後専用スレッドと呼ぶ。）を生成する。クラスライブラリ部による専用スレッドの生成はアプリケーションを実行しているスレッドで実行される。クラスライブラリ部は、アプリケーションを実行しているタスクに属するスレッドとして、リスナーを呼び出す専用スレッドを生成する。

【0051】ここで、リスナーとはイベント（ユーザーによってリモコン入力操作がされたことやモデムが相手先から切断されたことなどによるデバイスの状態変化）が発生したことの連絡を待ち受けるクラスまたはインターフェースであり、リスナーには、イベントがあった時に呼び出されるメソッドが定義され、当該メソッドを呼び出し、メソッドを実行することによりイベントの処理が行われる。登録されたリスナーは、イベント発生時に呼び出される。例えば、ユーザーによるリモコン操作入力等のイベントがあったときに登録されたリスナーに定義されたメソッドが呼び出され、呼び出されたメソッドを実行することによってイベントが処理される。また、「リスナー登録」とは、アプリケーションが「イベントが発生すれば、当該アプリケーションに通知してくれるように」当該イベントを監視するクラスライブラリ部に依頼することをいう。上記の通知は、当該クラスライブラリ部がリスナー登録時に当該リスナーに定義されるメソッドを呼び出すことにより実行される。

【0052】また、クラスライブラリ部はリスナー登録される際、アプリケーション管理部33bを呼び出し、リスナー登録をするアプリケーションのIDを取得する。アプリケーション管理部33bは、以下の処理を行うことによりアプリケーションのIDを特定し、クラスライブラリ部に特定したIDを提供する。アプリケーション管理部33bは呼び出しを行ったスレッドのスレッドクラスのインスタンスを取得し、当該インスタンスからインスタンスが保持するハッシュコード値を取り出すメソッドであるhashcode関数を用いて、呼び出しを行ったインスタンスのID（スレッドID）を特定し、アプリケーション管理部33bが管理するアプリケーションIDとスレッドIDの対応関係を示すテーブルから、スレッドIDに対応す

るアプリケーションIDを特定し、クラスライブラリ部に特定したIDを提供する。ここで、取り出されるハッシュコード値は、VM部３３aが効率よく、クラスから生成されたインスタンスを管理できるように、各インスタンスに割り振った識別番号（ID）である。各クラスライブラリ部は、アプリケーション管理部３３bから提供されたアプリケーションIDと生成した専用スレッドのIDと登録されたリスナーのIDとを対応付けてテーブルに保持する。

【００５３】図１８は、クラスライブラリ部が行うリスナー管理の処理を示すフローチャートである。クラスライブラリ部は、VM部３３aが実行するアプリケーションからリスナー登録のために呼び出されると、アプリケーション管理部３３bを呼び出し、リスナー登録をするアプリケーションのIDを取得し（S501）、登録されたリスナー、専用スレッド、アプリケーションIDの対応関係を示すテーブルから、アプリケーションIDと専用スレッドとリスナーの組の１つを取り出し（S502）、当該アプリケーションIDと取得したアプリケーションIDを比較する（S503）。

【００５４】IDが一致する場合（S503:Y）は、当該IDに対応付けて新たにリスナー登録されたリスナーをテーブルに追加する（S504）。IDが不一致の場合（S503：N）は、テーブルに保持されている次の組を取り出し（S505、S502）、同様の処理を繰り返す。IDが一致する場合がなく、IDの比較を終了すると（S505：Y）、新たに登録されたリスナーに対応する専用スレッドを生成し（S506）、取得したアプリケーションIDと生成した専用スレッドと新たに登録されたリスナーの組をテーブルに追加し、保持する（S507）。

【００５５】これにより、テーブルに保持されたアプリケーションが複数回リスナー登録をする場合に、アプリケーションに対応する専用スレッドをテーブルから特定することができ、アプリケーションによって登録された複数のリスナーを、特定した１つの専用スレッドに対応させてテーブルにまとめて保持することができる。従って、1つの専用スレッドから対応する複数のリスナーを呼び出すことができるので、アプリケーションからリスナー登録される毎に専用のスレッドを生成する必要がなく、生成するスレッドの数を少なくすることができる。

【００５６】図１９は、クラスライブラリ部が保持するテーブルの例である。図１９－（１）は、アプリケーションID「１」のアプリケーションがリスナー「L1」とリスナー「L4」を登録し、これらのリスナーに対応する専用スレッド「thread10」が生成されていることを示す。図１９－（２）は、図１９－（１）のテーブルにリスナー「L6」が登録されて、テーブルが更新されたことを示す。具体的には、リスナー「L6」がリスナー登録されると、クラスライブラリ部は、アプリケーション管理部３３bを呼び出して、リスナー登録をしたアプリケーションのアプリケーションIDを取得し（ここでは、ID「２」を取得したものとする。）、図１９－（１）のテーブルに保持されるアプリケーションID「１」と一致するか否かを判定し、一致しないと判定すると、新規に専用スレッドを生成し、生成した専用スレッド「thread20」とアプリケーションID「２」とリスナー「L6」とを対応させてテーブルに保持する。図１９－（３）は、アプリケーションID「１」のアプリケーションが終了し、テーブルに保持されていたアプリケーションID「１」と対応する専用スレッド「thread10」、リスナー「L1」とリスナー「L4」とがテーブルから削除されたことを示す。

【００５７】次にイベントが発生してから、クラスライブラリ部が専用スレッドから登録されたリスナーを呼び出すまでに行われる処理について説明する。この処理は、２つの処理から構成される。最初に行われる処理は、イベントが発生したことを通知する処理であり、この処理は、OS部３４により生成されたスレッドにより実行される。図２０は、イベントが発生したことを通知する処理を示すフローチャートである。クラスライブラリ部は、OS部３４のライブラリ格納部３４bに含まれるライブラリ部からイベントがあったことを通知されると、アプリケーションIDと専用スレッドと登録されたリスナーとの組を保持したテーブルから組の１つを取り出し（S601）、取り出した組の専用スレッドのキューにリスナー呼び出しに必要な呼び出し情報を保持する（S602）。ここで、キューは、スレッド間の情報伝達を行い、「情報を１つ保持する」、「情報を１つ取り出す」という２つの操作をすることができる。同様の処理をテーブルに保持する全ての組について行う（S601～S603）。

【００５８】次に行われる処理は、登録されたリスナーを呼び出す処理であり、この処理は、クラスライブラリ部が生成した専用スレッドにより実行される。図２１は、クラスライブラリ部が行うリスナー呼び出しの処理を示すフローチャートである。リクラスライブラリ部は、専用スレッドを実行させることにより、キューに呼び出し情報が保持されているか否かを監視し（S701）、キューに呼び出し情報があれば(S701:Y)、それを取り出し（S702）、テーブルから専用スレッドに対応するリスナーを特定し（S703）、リスナーを呼び出す（S704）。専用スレッドに対応するリスナーが複数ある場合は、このリスナー呼び出し処理（S703、S704）を対応する全てのリスナーについて行う（S703～S705）。図２２は、キューを用いて専用スレッドからリスナーを呼び出す手順の例を模式化した図である。

【００５９】図２２において、４１１、４１２はそれぞれ、図１９の専用スレッド「thread10」と専用スレッド「thread20」に対して用意されたキューを示す。また、４０１、４０２はそれぞれ、専用スレッド「thread10」と専用スレッド「thread20」を示し、４０３はOS部３４

のライブラリ格納部３４bに格納されるライブラリ部からイベントが発生したことを通知するOS部３４により生成されたスレッドであり、破線４２１、４２２は情報を保持させることを示し、黒丸４３１、４３２は保持された情報を示し、破線４４１、４４２は保持された情報を取り出すことを示す。

【００６０】クラスライブラリ部は、ライブラリ部からイベントが発生したことをスレッド４０３により通知されると、リスナー呼び出しに必要な情報（４３１、４３２）をキュー４１１、４１２に保持させる（４２１、４２２）。専用スレッド４０１、４０２は、キュー４１１、４１２に情報が保持されているか否かを監視し、情報が保持されていると、その情報を取り出し対応するリスナーを呼び出す。すなわち、専用スレッド４０１はリスナー「L1」とリスナー「L4」を、専用スレッド４０２はリスナー「L6」を呼び出す。

【００６１】OS部３４は、カーネル３４aとライブラリ格納部３４bから構成され、Javaミドルウェア部３３を動作させることにより、アプリケーションを間接的に動作させる。カーネル３４aは、アプリケーションに提供された資源をアプリケーションに対応するタスク単位で管理する。具体的には、カーネル３４aは、アプリケーション管理部３３bからの指示により、アプリケーションに対応するタスクとタスクに属する最初のスレッドを生成し、最初のスレッドでVM部３３aの動作に必要な資源を確保する。またアプリケーションが要求するスレッドを生成し、生成したスレッドでアプリケーションが要求する資源を提供し、提供した資源の資源名をスレッドが属するタスクIDに対応付けて保持する。アプリケーション管理部３３aから終了したアプリケーションに対応するタスクIDの通知を受けると、通知されたタスクIDに対応づけて保持された資源名の資源を全て回収する。

【００６２】ライブラリ格納部３４bは、資源であるデバイスを制御する複数のライブラリ部３４b1～３４bnからなる。各ライブラリ部は、ハードウェア部３５の対応するデバイスを制御する（例えばライブラリ部３４b1は、デバイス３５b1を、ライブラリ部３４b2は、デバイス３５b2を制御する。）。実際にはライブラリ格納部３４bは、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の関数群を記述したプログラムがクラスライブラリのクラスに呼び出されて実行されることによりデバイスの制御が実現されている。

【００６３】ハードウェア部３５は、OS部３４やJavaミドルウェア部３３を動作させるCPU３５a及びデバイス３５ｂ１～３５ｂｎで構成される。CPU３５aは、カーネル３４aが生成したスレッドを実行することによりアプリケーションのプログラムコードを実行する。デバイス３５b１～３５ｂｎは、メモリーデバイス ディスプレイデバイス、入力デバイス（キーボード、リモコン等）、ファイルシステム、ネットワークデバイスなどを含み、各デバイスはクラスライブラリ及びライブラリ部を介してなされるアプリケーションからの制御に応じて作動する。

【００６４】指示通知部３６は、アプリケーションの開始及び終了をアプリケーション管理部３３bに指示する。具体的には、アプリケーション管理部３３bにアプリケーション開始の指示をし、外部からアプリケーション終了の指示を受けると、アプリケーション管理部３３bにアプリケーションの終了を指示する。以上、本発明の実施の形態について説明したが、本発明はこの実施の形態に限定されないのは言うまでもない。

【００６５】なお、図２１のフローチャートでは、クラスライブラリは専用スレッドを実行させて、常にキューに呼び出し情報が保持されているか否かを監視する（S701）としたが、S701において、専用スレッドをウエイト状態にして、キューに呼び出し情報が保持されたときに、専用スレッドのウエイト状態を解除し、専用スレッドを実行させて、S702～S705の処理を行わせることとしてもよい。これにより、資源を効率的に利用することができる。

【００６６】なお、図１７のS801とS802は順序を逆にしてもよい。また、アプリケーション毎に生成されるスレッドの数は図１６の例に限らない。この例より多くても少なくてもよい。また、本実施の形態において、資源の管理は、カーネル３４aが行うとしたが、ライブラリ格納部３４bに含まれる各ライブラリ部が行うこととしてもよい。この場合、資源供給の際にカーネル３４aに資源を供給するタスクを問い合わせ、カーネル３４aより当該タスクのIDの通知を受けると、通知を受けたタスクIDと供給する資源の資源名を組にして保持し、カーネル３４aより特定のタスクIDの資源回収の指示があると、指示されたタスクIDに対応する資源名を保持した組から特定し、特定した資源名の資源を回収する。また、本実施の形態においては、VM部３３aは、アプリケーションを実行するタスクに含まれる最初のスレッドでVM部３３aに必要な資源が確保されるとしたが、アプリケーションを実行するスレッドから独立したシステム専用スレッドをクラスライブラリ部に生成させ、生成させたシステム専用スレッドで必要な資源を確保することとしてもよい。この場合のクラスライブラリ部の行う処理を図２３を用いて説明する。図２３は、VM部３３aがシステム専用のスレッドを獲得する処理を示すフローチャートである。クラスライブラリ部は、アプリケーションを実行するスレッドで、実行すべき処理（例えばアプリケーションが要求する資源を提供）し（S5401）、次にシステム専用のスレッドを生成し（S5402）、生成したシステム専用スレッドで実行すべき処理(例えばVM部３３aに必要な資源を提供する処理)を実行し（S5403）、処理が終了すると処理結果をアプリケーションに通知する（S5404）。アプリケーションを実行するスレッドとシステム

専用のスレッドの処理は、任意の回数、交互に繰り返し実行してもよいし、並行して実行してもよい。また、S5404の通知は、不要としてもよい。

【００６７】これにより、VM部３３aが必要とする資源は、システム専用のスレッドで確保されるので、VM部３３aはアプリケーションが終了しても自己に必要な資源を再取得することなく、次のアプリケーションを実行することができる。また、上記のシステム専用スレッドでクラスライブラリ部が自己に必要な資源を確保することとしてもよい。例えば、図１９に例として示すアプリケーションIDとスレッドインスタンスとリスナーインスタンスの対応関係を示すテーブルを保持するのに必要なメモリを上記システム専用スレッドで確保することとしてもよい。

【００６８】また、本実施の形態においては、クラスライブラリ部が専用のスレッドを生成するとしたが、アプリケーション実行中に画面に描画するための機能を提供するComponentクラスからComponentインスタンスを生成する時にも生成したインスタンスに格納されるpaintメソッドを呼び出すための呼び出し用スレッドを生成することとしてもよい。さらに、アプリケーションによるリスナー登録のためのクラスライブラリ部による専用スレッドの生成をアプリケーション起動時に完了することとしてもよい。

【００６９】これにより、クラスライブラリ部はリスナー登録する毎に専用スレッドの有無を確認する必要がなくなり、リスナー登録の処理を短時間で完了することができる。

（実施の形態４）本実施の形態におけるアプリケーション実行装置においては、各ライブラリ部がアプリケーションに提供する資源の回収を行い、アプリケーションが終了等する毎にアプリケーションが使用した資源を回収する。

【００７０】具体的には、以下の処理を行う。各ライブラリ部は各クラスライブラリ部を介してアプリケーションから資源の提供を要求されると、要求された資源を提供し、各クラスライブラリ部に資源を要求したアプリケーションのIDを取得し、取得したアプリケーションIDと提供した資源の資源名とを組にしてテーブルに保持する。ライブラリ部は、カーネル４４aより終了したアプリケーションIDの通知を受けると、通知されたアプリケーションIDに対応する資源名を、保持しているテーブルから特定し、特定した資源名の資源を全て回収する。

【００７１】以下本アプリケーション実行装置の実施の形態について、図面を用いて説明する。図２４は、アプリケーション実行装置４０の構成を示すブロック図である。アプリケーション実行装置４０は、アプリケーション入力部１１、アプリケーション記憶部１２、Javaミドルウェア部４３、OS部４４、ハードウェア部４５、指示通知部４６から構成される。

【００７２】アプリケーション入力部１１、アプリケーション記憶部１２は、実施形態１の実行装置１０に含まれるものと同一であるので、説明を省略する。Javaミドルウェア部４３は、VM部４３a、アプリケーション管理部４３b、クラスライブラリ格納部４３cから構成され、アプリケーション記憶部４２が記憶するアプリケーションを実行する。

【００７３】VM部４３aはアプリケーション記憶部１２からアプリケーションをロードし、アプリケーションのバイトコードを逐次解析し、アプリケーションを実行する。アプリケーション管理部４３bは、VM部４３bが実行するアプリケーションの起動の処理、実行中のアプリケーションの管理、アプリケーション終了の処理を行う。

【００７４】はじめにアプリケーション起動の処理について説明する。アプリケーション管理部４３bは、クラスローダのインスタンスをアプリケーション毎に生成し、クラスローダのインスタンスを用いてアプリケーションを構成する全てのクラスファイルをロードする。ここで、クラスローダは、Javaプログラムで使用されるクラスファイルをメモリに読み込むクラスである。

【００７５】次に実行中のアプリケーションの管理について説明する。アプリケーション管理部４３bは、起動したアプリケーションを構成するクラスファイルをロードするクラスローダのインスタンスと当該アプリケーションに割当てられたIDを組にしたテーブルを保持する。なお、IDはOS部４４のカーネル４４aによって、起動されたアプリケーション毎にそれぞれ異なるIDが割当てられる。図２５は、アプリケーション管理部４３bが保持するクラスローダのインスタンスと対応するアプリケーションIDを組にしたテーブルの例を示す。

【００７６】次にアプリケーション終了の処理について説明する。アプリケーション管理部４３bは、制御部４６からアプリケーション終了の指示を受けると、カーネル４４aとクラスライブラリ格納部４３cが格納するクラスライブラリ部に終了するアプリケーションIDを通知する。アプリケーション管理部４３bは、さらに資源をアプリケーションに提供したクラスライブラリ部から呼び出され、当該アプリケーションのアプリケーションID通知の要求があると、以下の処理を行って当該アプリケーションIDを特定し、クラスライブラリに通知する。

【００７７】アプリケーション管理部４３ｂは、クラスの呼び出し情報を格納したスタックを参照し、アプリケーションをロードしたクラスローダのインスタンスを取得し、保持しているテーブルから取得したクラスローダのインスタンスに対応するアプリケーションIDを特定し、特定したアプリケーションIDを当該クラスライブラリ部に通知する。ここで、スタックは、クラスのメソッドの作業領域として割当てられるメモリ領域であり、メソッドの呼び出しによりクラスの呼び出しが行われる毎に、呼び出されたクラスの作業領域がスタック中に割当

てられ、スタック中に呼び出し情報を保持するクラスが順番に格納される。呼び出し情報には、メソッドの呼び出し元を特定する情報が格納される。

【００７８】以下具体例を挙げて、アプリケーションIDを特定する処理について詳しく説明する。図２６はアプリケーション管理部４３bが生成したクラスローダのインスタンスとスタックとの関係を示す模式図である。ここでは、クラスローダのインスタンス６３０１がアプリケーションを構成するクラスAのメソッドを呼び出し、呼び出されたクラスAがアプリケーションを構成する別のクラスBのメソッドを呼び出し、次にクラスBがアプリケーションを構成する別のクラスCを呼び出し、最後にクラスCがクラスライブラリ部に格納するクラスDのメソッドを呼び出し、資源の確保を要求した場合の例を用いて説明する。この場合、スタック中には、呼び出しがされる毎に、呼び出されたクラスの作業領域が割当てられ、割当てられた作業領域に呼び出し情報が格納される。６３００は、アプリケーション管理部４３ｂが管理するクラスローダの格納場所を表し、ここでは、２つのクラスローダ６３０１と６３０２が格納されている。６３１０は、スタックを表し、６３１１～６３１４はスタック６３１０に格納されているクラスA～Dに保持された呼び出し情報を示す。アプリケーション管理部４３ｂは、クラスDに呼び出されたクラスライブラリ部からアプリケーションIDの特定要求があると、スタック６３１０に格納された呼び出し情報６３１４～６３１１を順に参照することにより、アプリケーションをロードしたクラスローダのインスタンス６３０１を取得する。アプリケーション管理部４３bは、保持しているテーブルを参照し、取得したクラスローダのインスタンスに対応するアプリケーションIDを特定する。

【００７９】クラスライブラリ格納部４３cは、複数のクラスライブラリ部４３c1～４３ｃｎからなり、各クラスライブラリ部は、ライブラリ格納部４４bに含まれる対応する各ライブラリ部を介して、ハードウェア部４５に含まれる資源である各デバイスを制御する。実際には、クラスライブラリ格納部４３cは、ROMやハードディスクなどで構成され、各クラスライブラリ部では、各デバイス制御用のプログラムであるクラス群に記述されたプログラムがアプリケーションに呼び出されて実行されることにより各デバイスの制御が実現されている。

【００８０】各クラスライブラリ部は、アプリケーションから呼び出され、資源であるデバイスの提供を要求されると、ライブラリ格納部４４bに含まれる当該デバイスに対応するライブラリ部を呼び出し、要求された資源を当該アプリケーションに提供する。さらに、クラスライブラリ部は当該アプリケーションのアプリケーションIDをアプリケーション管理部４４bから取得し、当該ライブラリ部に取得したアプリケーションIDを通知する。

【００８１】カーネル４４aは、アプリケーション管理部４３bからアプリケーション終了の通知と終了したアプリケーションのアプリケーションIDの通知を受け、各ライブラリ部に受け取ったアプリケーションIDを通知し、資源の回収を指示する。ライブラリ格納部４４bは、デバイスを制御する複数のライブラリ部４４b1～４４bｎからなる。各ライブラリ部は、ハードウェア部４５の対応するデバイスを制御する（例えばライブラリ部４４b1は、デバイス４５b1を、ライブラリ部４４b2は、デバイス４５b2を制御する。）。

【００８２】実際にはライブラリ格納部４４bは、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の関数群を記述したプログラムが各クラスライブラリ部のクラスに呼び出されて実行されることにより資源であるデバイスの制御が実現されている。具体的には、各ライブラリ部はクラスライブラリ部からの呼び出しに応じて、アプリケーションに対して資源を提供し、クラスライブラリ部から当該アプリケーションのアプリケーションIDの通知を受け、提供した資源の資源名と受け取ったアプリケーションIDを組にしてテーブルに保持する。カーネル４４aより終了したアプリケーションのアプリケーションIDの通知を受けると、保持しているテーブルから通知を受けたアプリケーションIDに対応する資源名を特定し、特定した資源名の資源を回収する。図２７は、提供した資源の資源名とアプリケーションIDを組にしたテーブルの一例である。ここでは、ライブラリ部が２つのアプリケーションに資源（ファイル）を提供していることを示す。

【００８３】なお、本実施形態においては、各ライブラリ部が提供した資源の資源名とアプリケーションIDを組にしたテーブルを保持することとしたが、カーネル４４aがテーブルを保持することとしてもよい。また、本実施形態においては、カーネル４４aがアプリケーションIDを生成することとしているので、複数のミドルウェアを同時に動作させる場合においても、２つのミドルウェアがそれぞれ個別にIDを生成することにより生じるIDの重複を防ぐことができ、カーネル４４aが正しく資源を管理することができる。

(実施の形態５)図２８は、本発明の実施の形態におけるアプリケーション実行装置１００の構成を示すブロック図である。アプリケーション実行装置１００は、アプリケーション/クラスライブラリ記憶部１０１、アプリケーション管理部１０２、VM（Virtual Machine）部１０３、メモリヒープ管理部１０４、メモリ１０５を備える。ここで、メモリヒープ管理部１０４は、オブジェクト領域獲得部１０４a、分割ヒープ領域獲得部１０４b、分割ヒープ領域解放部１０４c、GC部１０４d、ロック部１０４e、メモリヒープ領域管理テーブル１０４ｆを備える。

【００８４】メモリ１０５は、複数の分割ヒープ領域１０５ａ～１０５ｎを含むメモリヒープ領域を有する。各

分割ヒープ領域は、１つのアプリケーションに対応して設けられ、対応するアプリケーションに関連するオブジェクトを保持するための領域である。アプリケーション個別に分割ヒープ領域が設けられるのは、１回のガベージコレクションの対象領域をメモリヒープ領域全体ではなく分割ヒープ領域単位にするためである。加えて、アプリケーション終了時にはメモリヒープ領域は分割ヒープ領域単位に解放される。

【００８５】アプリケーション/クラスライブラリ記憶部１０１は複数のアプリケーション及び既存のクラスライブラリを記憶する。アプリケーション管理部１０２は、外部からアプリケーションの起動及び終了の指示を受け、ＶＭ部１０３に当該アプリケーションの起動及び終了を指示する。また、アプリケーション管理部１０２は、アプリケーションが終了すると、分割ヒープ領域解放部１０４ｃに当該アプリケーションに関連するオブジェクトが配置された分割ヒープ領域の解放を指示する。

【００８６】ＶＭ部１０３は、アプリケーション管理部１０２からアプリケーションの起動指示を受けると、当該アプリケーション用に分割ヒープ領域の割り当てをメモリヒープ管理部１０４に行なわせ、割り当てられた分割ヒープ領域にアプリケーション／クラスライブラリ記憶部１０１から当該アプリケーションをロードするとともにロードしたアプリケーションを実行する。

【００８７】より詳しく説明すると、アプリケーションのロードは、第１にクラスローダオブジェクトのロード、第２にアプリケーションに関連する個々のオブジェクトのロードに分けられる。まず、クラスローダオブジェクトのロードについて説明する。ＶＭ部１０３は、アプリケーション管理部１０２から上記アプリケーション起動指示として、当該アプリケーション用のクラスローダオブジェクトの生成指示を受け、当該クラスローダオブジェクト用のオブジェクト領域の獲得指示（以下、第１獲得指示と呼ぶ。）をオブジェクト領域獲得部１０４ａに出力する。ここで、クラスローダオブジェクトとは、対応するアプリケーションに関連するオブジェクトをアプリケーション／クラスライブラリ記憶部１０１から分割ヒープ領域にロードするためのオブジェクトである。

【００８８】ＶＭ部１０３は、第１獲得指示に対して、オブジェクト領域獲得部１０４ａから、分割ヒープ領域の割り当て及びオブジェクト領域の割り当てを受けると、アプリケーション／クラスライブラリ記憶部１０１から当該オブジェクト領域にクラスローダオブジェクトをロードして当該クラスローダオブジェクトを実行する。

【００８９】次に、アプリケーションに関連する個々のオブジェクトのロードについて説明する。ＶＭ部１０３は、このクラスローダオブジェクトを実行することにより、対応するアプリケーションに関連する個々のオブジェクトをアプリケーション／クラスライブラリ記憶部１０１からオブジェクト領域獲得部１０４aによって割り当てられたオブジェクト領域にロードする。すなわち、ＶＭ部１０３は、対応するアプリケーションに関連するオブジェクトについてオブジェクト領域の獲得指示（以下、第２獲得指示と呼ぶ）をオブジェクト領域獲得部１０４ａに出力し、第２獲得指示に対する応答として、オブジェクト領域獲得部１０４ａからオブジェクト領域の割り当てを受けると、アプリケーション／クラスライブラリ記憶部１０１から当該オブジェクト領域にオブジェクトをロードして当該オブジェクトを実行する。上記第２獲得指示は、クラスローダオブジェクト以外のオブジェクトに対するオブジェクト領域の獲得指示であって、実行中のアプリケーションから他のオブジェクトの生成要求があればＶＭ部１０３から出力される。

【００９０】オブジェクト領域獲得部１０４aは、VM部１０３からの第１獲得指示を受けると、分割ヒープ領域獲得部１０４ｂに分割ヒープ領域の獲得を指示して分割ヒープ領域の割り当てを受けると、当該分割ヒープ領域内にクラスローダオブジェクト領域を獲得し、当該クラスローダオブジェクト領域をＶＭ部１０３に通知する。

【００９１】メモリヒープ管理テーブル１０４ｆは、分割ヒープ領域獲得部１０４ｂによって獲得された分割ヒープ領域に関する情報（以下分割ヒープ領域情報と呼ぶ。）から構成される。図２９に示すように各分割ヒープ領域情報は、分割ヒープ領域のID、分割ヒープ領域にロードされるクラスローダオブジェクトのID、分割ヒープ領域の先頭アドレス、分割ヒープ領域全体のサイズ、分割ヒープ領域中の未使用領域を示す自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズから構成される。メモリヒープ管理テーブル１０４ｆへの分割ヒープ領域情報の新規登録は、分割ヒープ領域獲得部１０４ｂがオブジェクト領域獲得部１０４aからの分割ヒープ領域の獲得の指示を受けて分割ヒープ領域を獲得した時に、分割ヒープ領域獲得部１０４ｂによって行われる。　上記の分割ヒープ領域情報中の自由領域の先頭アドレス及び分割ヒープ領域中の自由領域のサイズに関する情報は、VM部１０３からの第１又は第２獲得指示によってオブジェクト領域獲得部１０４aがオブジェクト領域を獲得する毎に更新される。又アプリケーション終了後、アプリケーション管理部１０２の指示に応じて分割ヒープ領域解放部１０４ｃが終了したアプリケーションに割当てられた分割ヒープ領域を解放した際に、当該分割ヒープ領域に関する分割ヒープ領域情報はメモリヒープ管理テーブル１０４ｆから削除される。

【００９２】また、オブジェクト領域獲得部１０４a は、VM部１０３からの第２獲得指示を受けると、当該第２獲得指示の要求元アプリケーションをロードしたクラスローダオブジェクトを特定し、特定したクラスローダオブジェクトに対応する分割ヒープ領域をメモリヒープ

領域管理テーブル１０４ｆを参照して特定（例えば特定したクラスローダがCL2である場合には、分割ヒープ領域ｂを特定）し、特定した分割ヒープ領域内でオブジェクト領域の獲得を試みる。オブジェクト領域獲得部１０４aは特定した分割ヒープ領域からオブジェクト領域を獲得できた場合、当該オブジェクト領域をＶＭ部１０３に通知し、メモリヒープ管理テーブル１０４ｆの当該分割ヒープ領域の情報（当該分割ヒープ領域中の自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズ）を更新する。

【００９３】一方、オブジェクト領域獲得部１０４aは、当該分割ヒープ領域からオブジェクト領域を獲得できなかった場合は、GC部１０４ｄに対し、当該分割ヒープ領域のガベージコレクションを指示し、ガベージコレクション終了後に再度オブジェクト領域の獲得を試みる。分割ヒープ領域獲得部１０４bは、オブジェクト領域獲得部１０４aからの分割ヒープ領域獲得指示を受けると、メモリヒープ領域から分割ヒープ領域を獲得する。分割ヒープ領域獲得部１０４ｂは、獲得した分割ヒープ領域１０５aの情報をクラスローダオブジェクトのIDと組にしてヒープメモリ領域管理テーブル１０４ｆに登録する。

【００９４】分割ヒープ領域解放部１０４cは、アプリケーション管理部１０２からアプリケーションの終了を指示されると、アプリケーション管理部１０２からの分割ヒープ領域の解放指示に応じて、終了したアプリケーションをロードしたクラスローダオブジェクトに対応する分割ヒープ領域をメモリヒープ領域管理テーブル１０４ｆから特定し、特定した分割ヒープ領域を解放し、解放した分割ヒープ領域の情報（解放した分割ヒープ領域のID、当該分割ヒープ領域に対応するクラスローダオブジェクトのID、当該分割ヒープ領域の先頭アドレス、当該分割ヒープ領域全体のサイズ、当該分割ヒープ領域中の自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズとそれに対応するクラスローダオブジェクトのID）をメモリヒープ領域管理テーブル１０４ｆから削除する。

【００９５】GC部１０４ｄは、オブジェクト領域獲得部１０４aの指示に応じて分割ヒープ領域のガベージコレクションを行う。ロック部１０４eは、GC部１０４ｄが分割ヒープ領域のガベージコレクションを行っている間、当該分割ヒープ領域を使用するアプリケーションの実行を停止する。

【００９６】図３０は、VM部１０３よりオブジェクト領域獲得指示（第１獲得指示又は第２獲得指示）を受けた後のアプリケーション実行装置１００によるオブジェクト領域獲得動作を示すフローチャートを示す。オブジェクト領域獲得部１０４aは、VM部からのオブジェクト領域獲得指示が第１（クラスローダオブジェクト）獲得指示か第２（クラスローダ以外のオブジェクト）獲得指示かを判定する（S1301）。

【００９７】第１獲得指示の場合（S1301:Y）、オブジェクト領域獲得部１０４aは、当該クラスローダオブジェクトに対応する分割ヒープ領域の獲得を分割ヒープ領域獲得部１０４ｂに要求し、分割ヒープ領域獲得部１０４ｂは、メモリ１０５から分割ヒープ領域を獲得する（S1302）。分割ヒープ領域獲得部１０４ｂは、図２９の例に示す通り、メモリヒープ領域管理テーブル１０４ｆに当該クラスローダオブジェクトのIDと獲得した分割ヒープ領域の情報を組にして登録する（S1303）。次にオブジェクト領域獲得部１０４aは、分割ヒープ領域獲得部１０４ｂが獲得した分割ヒープ領域内に当該クラスローダオブジェクトのオブジェクト領域を獲得し（S1306、S1307）、当該分割ヒープ領域に対応するメモリヒープ領域管理テーブル１０４ｆの情報を更新する（S1309）。

【００９８】第２獲得指示の場合（S1301:N）、オブジェクト領域獲得部１０４aは、VM部１０３から当該オブジェクトが保有するクラスローダIDを取得し（S1304）、メモリヒープ領域管理テーブル１０４ｆから上記クラスローダIDに対応する分割ヒープ領域を特定する（S1305）。次に、オブジェクト領域獲得部１０４aは、特定した分割ヒープ領域内にオブジェクト領域を獲得する（S1306）。オブジェクト領域獲得部１０４aは、オブジェクト領域の獲得に成功したか否かを判定する（S1307）。

【００９９】オブジェクト領域獲得部１０４aは、オブジェクト領域の獲得に成功した場合（S1307:Y）、当該分割ヒープ領域に対応するメモリヒープ領域管理テーブル１０４ｆの情報を更新する（S1309）。オブジェクト領域獲得部１０４aは、オブジェクト領域の獲得に失敗した場合（S1307:N）、GC部１０４ｄに獲得に失敗した分割ヒープ領域内のガベージコレクションを行うことを指示する。GC部１０４ｄは指示に応じて当該分割ヒープ領域のガベージコレクション処理を行う（S1308）。GC部１０４ｄがガベージコレクション処理を行っている間、ロック部１０４eは、当該分割ヒープ領域を使用するアプリケーションの実行を停止する。オブジェクト領域獲得部１０４aはガベージコレクション処理後、再度当該分割ヒープ領域からオブジェクト領域を獲得し（S3306）、オブジェクト領域の獲得に成功した場合（S1307:Y）、当該分割ヒープ領域に対応するメモリヒープ領域管理テーブル１０４ｆの情報を更新する（S1309）。

【０１００】このように、アプリケーション管理装置１００においては、生成したオブジェクトを配置するためのオブジェクト領域をアプリケーション毎に分割したメモリサイズの小さい分割ヒープ領域として割当てるため、GC部１０４ｄによる１回のガベージコレクション処理の負荷を軽減することができる。又、VM部１０３が複数のアプリケーションを実行させている場合、ロック部

１０４ｆは分割ヒープ領域のガベージコレクション処理の実行中、当該分割ヒープ領域を使用するアプリケーションだけを停止させるため、VM部１０３はその他の分割ヒープ領域を使用しているアプリケーションを引き続き実行させることができる。

【０１０１】図３１は、アプリケーション終了時のアプリケーション実行装置１００による分割ヒープ領域の解放の動作を示すフローチャートである。アプリケーション管理部１０２は、アプリケーションが終了すると、分割ヒープ領域解放部１０４ｃに、終了したアプリケーションをロードしたクラスローダオブジェクトに対応する分割ヒープ領域の解放を指示する（S1401）。分割ヒープ領域解放部１０４ｃは、メモリヒープ領域管理テーブル１０４ｆから当該アプリケーションをロードしたクラスローダオブジェクトに対応する分割ヒープ領域を特定し（S1402）、その情報に基いて当該分割ヒープ領域を解放する（S1403）。分割ヒープ領域解放部１０４ｃは、解放した分割ヒープ領域の情報と対応するクラスローダオブジェクトのIDをメモリヒープ領域管理テーブル１０４ｆから削除する（S1404）。これにより、アプリケーション実行装置１００においては、アプリケーション終了する毎に、対応する分割ヒープ領域を一括して、短時間で解放することができる。

【０１０２】なお、本実施形態においては、分割ヒープ領域獲得部１０４ｂは、システムクラスローダオブジェクトを意識することなく、分割ヒープ領域をシステムクラスローダオブジェクト用の分割ヒープ領域として割当てているが、図３２に示すようにメモリ１０５全体をシステムクラスローダ用のメモリ領域（システムヒープ領域１０５－１）として割当て、その領域内に分割ヒープ領域獲得部１０４ｂが他のクラスローダオブジェクトに対応する分割ヒープ領域を獲得するようにしてもよい（図３２）。この場合、オブジェクト領域獲得部１０４ａがシステム起動時にシステムクラスローダオブジェクトのオブジェクト領域をシステムヒープ領域１０５－１内に獲得し、システムクラスローダオブジェクトのIDと獲得した領域（説明の便宜上、分割ヒープ領域１０５a0と呼ぶ。）の情報（分割ヒープ領域ID（a0）、クラスローダID（SCL）、領域先頭アドレス（MA-0）、全体領域サイズ（MS-0）、自由領域先頭アドレス（FA-0）、自由領域サイズ（FS-0））を組にしてメモリヒープ領域管理テーブル１０５ｆに登録する。以後本実施形態の場合と同様に、分割ヒープ領域獲得部１０４ｂがシステムヒープ領域１０５－１から分割ヒープ領域をクラスローダオブジェクト領域用に獲得する毎に、分割ヒープ領域獲得部１０４ｂは、図３２に示すようにメモリヒープ領域管理テーブル１０４ｆに当該クラスローダオブジェクトのIDと獲得した分割ヒープ領域の情報を組にして追加し、オブジェクト領域獲得部は図３０のフローチャートに従って分割ヒープ領域獲得部１０４ｂによって獲得された分割ヒープ領域内にオブジェクト領域を獲得し（S1306）、オブジェクト領域の獲得に失敗した場合（S1307）、オブジェクト領域獲得部１０４ｂは、GC部１０４ｄに獲得に失敗した分割ヒープ領域内のガベージコレクションを行うことを指示し、GC部１０４ｄは指示に応じて当該分割ヒープ領域内のガベージコレクションを行う（S1308）。また、アプリケーションが終了すると、アプリケーション管理部１０２は、分割ヒープ領域解放部１０４ｃに、終了したアプリケーションをロードしたクラスローダオブジェクトに対応する分割ヒープ領域の解放を指示し、分割ヒープ領域解放部１０４ｃは、指示に応じて、図３１のフローチャートに従って当該分割ヒープ領域を解放する。また、GC部１０４ｄはシステムヒープ領域１０５－１全体をガベージコレクションの対象とし、分割ヒープ領域を単位に解放された分割ヒープ領域同士をひとまとめにするので、システムヒープ領域１０５－１を効率的に使用することができる。

【０１０３】なお、本実施形態における図２８のアプリケーション実行装置１００の構成は、機能ブロック図として示されているが、実際はCPU、メモリ、外部との入出力を行う入出力装置を含むハードウエア及びソフトウエアを実行することにより実現される。このソフトウエアは図２８に示した各ブロックの機能を管理するように設計されている。また上記ソフトウエアを、CD-ROMからロードする場合はCD-ROM読み取りインターフェースを用いてもよい。メモリは、RAM、ROM、ハードディスクや、CDドライブ、フロッピーディスク、メモリカードなどのリムーバブルメディアで構成される。

【０１０４】本実施形態においては、アプリケーションに、割当てられる分割ヒープ領域の数は、１つに限らず複数であってもよい。例えば、アプリケーション自身がクラスローダを保有している場合には、当該クラスローダに対しても分割ヒープ領域が割当てられる。また、本実施形態において、分割ヒープ領域獲得部１０４ｂによって獲得される分割ヒープ領域の大きさは、分割ヒープ領域獲得部１０４ｂへのプログラム指示により可変とすることができる。

【０１０５】

【発明の効果】本発明は、アプリケーションに資源を提供する少なくとも１つ以上のライブラリ部とカーネル部を備えるアプリケーション実行装置であって、前記カーネル部は、アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知する通知手段を有し、各ライブラリ部は、通知手段から通知されると、終了したアプリケーションに提供した資源を回収する回収手段を有することを特徴とする。また、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよいし、前記回収手段は、アプリケー

ションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することとしてもよいし、前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する提供手段と、前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知を行うよう依頼する依頼手段とを有することとしてもよい。

【0106】この構成により、アプリケーションが終了する毎に、カーネル部ではなくライブラリ部により不必要になった資源が回収されるので、新規の資源が追加される毎にカーネル部を修正する必要がなくなるという効果が得られる。ここで、前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を有し、前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する提供手段と、前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知を行うよう当該アプリケーションに対応する通知部に依頼する依頼手段とを有し、前記依頼手段は、前記本ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、依頼された前記通知部は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行うこととしてもよい。

【0107】この構成により、終了したアプリケーションに資源を提供しているライブラリ部にのみ、終了したアプリケーションがどれであるかが通知されるので、当該資源を提供していないライブラリ部への無駄な通知をすることなく、当該資源を回収することができるという効果が得られる。ここで、前記回収手段は、アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有し、前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供したライブラリ部に通知し、前記資源特定手段は、通知された複数のアプリケーションに対応する複数の資源を前記テーブル保持手段により特定し、前記回収手段は、特定された複数の資源を回収することとしてもよい。

【0108】この構成により、ライブラリ部は、終了した複数のアプリケーションに提供した資源を一括して回収できるので、より効果的な資源の回収を行うことができる。ここで、前記通知手段は、さらにアプリケーションが終了した時と中断した時の何れかの時に、前記各ライブラリ部に通知し、前記各ライブラリ部はさらに、通知されたアプリケーションが終了したか、中断したかに応じて、当該アプリケーションに提供した資源を回収するか否かを判定する判定手段を有し、前記回収手段は、さらに資源を回収すると判定された場合に、通知されたアプリケーションに提供した資源を回収することとしてもよい。また、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0109】この構成により、アプリケーションが中断しているか、終了しているかに応じて、資源を回収すべきか否かを判定し、特定の一部の資源のみ回収することができる。また、本発明は、アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置であって、前記Javaミドルウェア部は、アプリケーションとアプリケーションに対応するタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに対応するタスクがどれであるかを通知する通知手段とを有し、前記OS部は、アプリケーション毎にアプリケーションを実行するタスクを生成するタスク生成手段と、前記タスクを構成するアプリケーション用スレッドを複数生成するスレッド生成手段と、前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を提供し、提供した資源と前記スレッドが属するタスクとの対応関係を示すテーブルを保持する管理手段と、前記通知手段から通知されたタスクに対応する資源を管理手段より特定し、アプリケーションに提供されていた資源を回収する回収手段とを有することを特徴としてもよい。また、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0110】この構成により、アプリケーションが終了する毎に不必要になった資源が回収されるので、Javaアプリケーションを連続して実行するための資源を確保できるという効果が得られる。ここで、前記Javaミドルウェア部はさらに、前記タスクを構成する前記アプリケーション用スレッドとは別に、前記Javaミドルウェア部に必要な資源を確保する資源確保用スレッドを生成する資源確保用スレッド生成手段と、生成した資源確保用スレッドを実行することにより前記Javaミドルウェア部に必要な資源を確保する資源確保手段とを有し、前記回収手段は、前記資源確保手段によって確保された前記Javaミドルウェア部に必要な資源を回収することなく、前記通知手段から通知されたタスクに対応する資源を前記管理

手段より特定し、アプリケーションに提供されていた資源を回収することとしてもよい。

【0111】この構成により、Javaミドルウェア部に必要な資源が確保され、アプリケーションが終了しても当該資源は回収されないので、より短い時間で新たなアプリケーションを実行させることができる。また、本発明はガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するアプリケーション実行装置であって、アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手段と、起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割当てる割当て手段と、アプリケーションが終了する毎に、当該アプリケーションに割当てられた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴としてもよい。また上記アプリケーション実行装置における各手段をステップとするメモリヒープ管理方法としてもよいし、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0112】この構成により、アプリケーションが終了する毎に、終了したアプリケーションに割当てられた分割ヒープ領域が解放され、解放された分割ヒープ領域の内部ではガベージコレクションが不要になるので、ガベージコレクションの負荷を軽減することができる。ここで、前記アプリケーション実行装置はさらに、アプリケーションに関連するオブジェクトに対して、当該アプリケーションに割当てられた分割ヒープ領域内でオブジェクト領域を獲得するオブジェクト領域獲得手段と、前記分割ヒープ領域を対象にガベージコレクションを行うガベージコレクション手段とを備えることとしてもよい。また、上記アプリケーション実行装置における各手段をステップとするメモリヒープ管理方法としてもよいし、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0113】この構成により、メモリヒープ領域全体よりもメモリサイズの小さい分割ヒープ領域を対象としてガベージコレクションを行うために、1回のガベージコレクションの負荷を軽減することができる。ここで、前記アプリケーション実行装置はさらに、前記ガベージコレクション手段によって対象となる分割ヒープ領域のガベージコレクションが行われている間、当該分割ヒープ領域を使用するアプリケーションのみ実行を停止させ、他の分割ヒープ領域を使用するアプリケーションは継続して実行させるロック手段を備えることとしてもよい。また、上記アプリケーション実行装置における各手段をステップとするメモリヒープ管理方法としてもよいし、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0114】この構成により、ガベージコレクションが行われている間も、当該処理の対象となっている分割ヒープ領域を使用するアプリケーション以外のアプリケーションの実行を継続させるので、ユーザーは当該処理が終わるのを待つことなく、他のアプリケーションを使った作業を続行することができる。

【図面の簡単な説明】

【符号の説明】

【図1】アプリケーション実行装置10の構成を示すブロック図である。

【図2】コールバック関数を定義したC言語プログラムの一例を示す。

【図3】コールバック関数の登録と呼び出し実行を記述したC言語プログラムの例を示す。

【図4】各ライブラリ部が行う資源提供の処理を示すフローチャートである。

【図5】ファイルシステムを制御するライブラリ部（以下ライブラリ部Fと呼ぶ。）が保持するアプリケーション識別子と資源名（ファイル名）とを組にしたテーブルを示す。

【図6】各ライブラリ部が行う資源の回収処理を示すフローチャートである。

【図7】複数のアプリケーションが一斉に終了する場合の各ライブラリ部が行う資源の回収処理のフローチャートを示す。

【図8】アプリケーション実行装置20の構成を示すブロック図である。

【図9】インターフェースを定義するJavaプログラムの例を示す。

【図10】アプリケーション情報インスタンスを生成するクラスの一例を示す。

【図11】資源回収クラスの一例を示す。

【図12】資源回収クラスが生成した資源回収インスタンスの例を示す。

【図13】各ライブラリ部が行う資源提供の処理を示すフローチャートである。

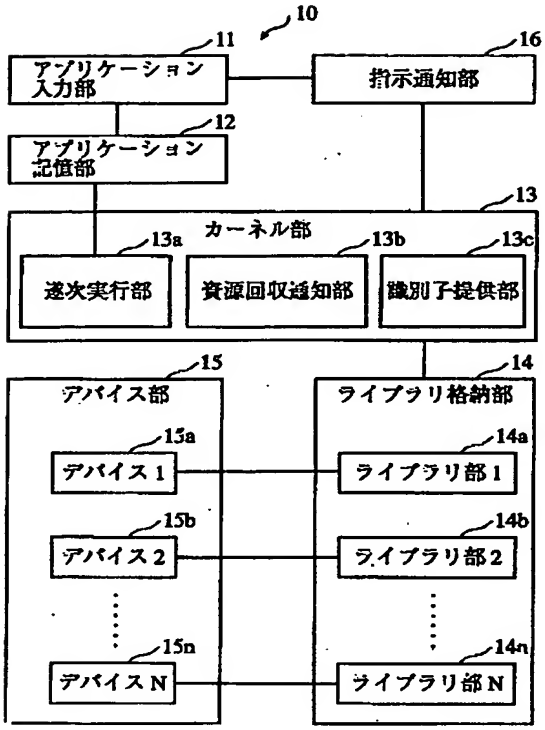
【図14】アプリケーション情報インスタンスがアプリケーションの状態を知るためのメソッドと資源回収インスタンスの登録を削除するメソッドを格納している場合の各ライブラリ部が行う回収の処理を示すフローチャートである。

【図15】アプリケーション実行装置30の構成を示すブロック図である。

【図16】アプリケーションとアプリケーションに対応して生成されたタスクとタスクを構成するスレッドとの対応関係を示すテーブルの例を示す。

【図17】アプリケーション終了時にアプリケーション管理部33bが行う処理を示すフローチャートである。
【図18】クラスライブラリ部が行うリスナー管理の処理を示すフローチャートである。
【図19】クラスライブラリ部が保持するテーブルの例である。
【図20】イベントが発生したことを通知する処理を示すフローチャートである。
【図21】クラスライブラリ部が行うリスナー呼び出しの処理を示すフローチャートである。
【図22】キューを用いて専用スレッドからリスナーを呼び出す手順の例を模式化した図である。
【図23】VM部33aがシステム専用のスレッドを獲得する処理を示すフローチャートである。
【図24】アプリケーション実行装置40の構成を示すブロック図である。
【図25】アプリケーション管理部43bが保持するクラスローダのインスタンスと対応するアプリケーションIDを組にしたテーブルの例を示す。
【図26】アプリケーション管理部43bが生成したクラスローダのインスタンスとスタックとの関係を示す模式図である。
【図27】提供した資源の資源名とアプリケーションIDを組にしたテーブルの一例を示す。
【図28】アプリケーション実行装置100の構成を示すブロック図である。
【図29】メモリヒープ領域管理テーブル104fとメモリ105の状態を示す図である。
【図30】VM部103よりオブジェクト領域獲得指示を受けた後のアプリケーション実行装置100によるオブジェクト領域獲得の動作を示すフローチャートである。
【図31】アプリケーション終了時のアプリケーション実行装置100による分割ヒープ領域の解放の動作を示すフローチャートである。
【図32】システムヒープ領域105－1がメモリヒープ領域管理テーブル104fに登録されている状態を示す図である。
【符号の説明】
1　メモリヒープ管理部
2　ヒープ領域獲得部
3　GC（ガベージコレクション）部
10　アプリケーション実行装置
11　アプリケーション入力部
12　アプリケーション記憶部
13　カーネル部
13a　逐次実行部
13b　資源回収通知部
13c　識別子提供部
14　ライブラリ格納部
14a～n　ライブラリ部
15　デバイス部
15a～n　デバイス
20　アプリケーション実行装置
23　OS部
23a　バーチャルマシーン部
23b　アプリケーション管理部
23c～e　アプリケーション情報インスタンス
24　ライブラリ格納部
24a～n　ライブラリ部
30　アプリケーション実行装置
33　Javaミドルウェア部
33a　VM部
33b　アプリケーション管理部
33c　クラスライブラリ格納部
33c1～c2　クラスライブラリ部
34　OS部
34a　カーネル
34b　ライブラリ格納部
34b1～b2ライブラリ部
35　ハードウェア部
35a　CPU
35b1～b2デバイス
40　アプリケーション実行装置
43　Javaミドルウェア部
43a　VM部
43b　アプリケーション管理部
43c　クラスライブラリ格納部
43c1～c2　クラスライブラリ部
44　OS部
44a　カーネル
44b　ライブラリ格納部
44b1～b2ライブラリ部
45　ハードウェア部
45a　CPU
45b1～b2デバイス
100　アプリケーション実行装置
101　アプリケーション/クラスライブラリ記憶部
102　アプリケーション管理部
103　VM部
104　メモリヒープ管理部
104a　オブジェクト領域獲得部
104b　分割ヒープ領域獲得部
104c　分割ヒープ領域解放部
104d　GC部
104e　ロック部
104f　メモリヒープ領域管理テーブル
105　メモリ
105a～105n　分割ヒープ領域

【図1】



【図2】

```
typedef void(*CALLBACK_f)(int Appid)
```

【図5】

(1)

アプリケーション識別子	ファイル識別子
1	a.txt

(2)

アプリケーション識別子	ファイル識別子
1	a.txt
2	b.txt

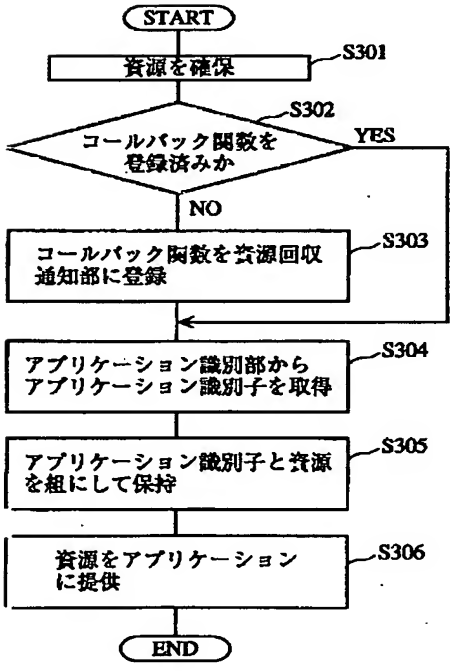
(3)

アプリケーション識別子	ファイル識別子
1	a.txt

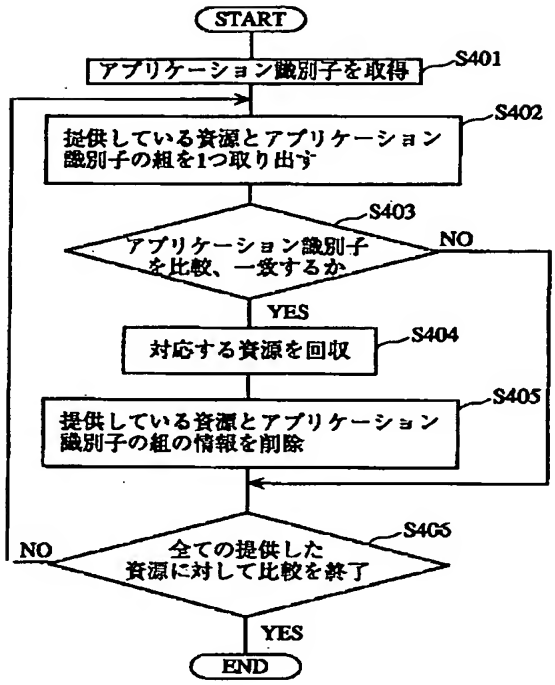
【図3】

```
1   CALLBACK_f callf[100];
2
3   void init() {
4     for (I-0;I<100; I++ ) callf[I]=null;
5   }
6   int add_callf(CALLBACK_f callfunc){
7     for (I=0; I<100; I++ ){
8       if ( callf[I]==null) {
9         callf[I] = callfunc;
10        return(1);
11      }
12    }
13    return(-1);
14  }
15
16  void remove_callf(CALLBACK_f callfunc){
17    for (I =0; I<100; I++){
18      if ( callf[I]==callfunc ) callf[I] = null;
19    }
20  }
21
22  void call_callback (int Appid) {
23    for(I =0; I<100; I++ ){
24      if ( (callf[I] !=null) (*callf[I])(Appid);
25    }
26  }
```

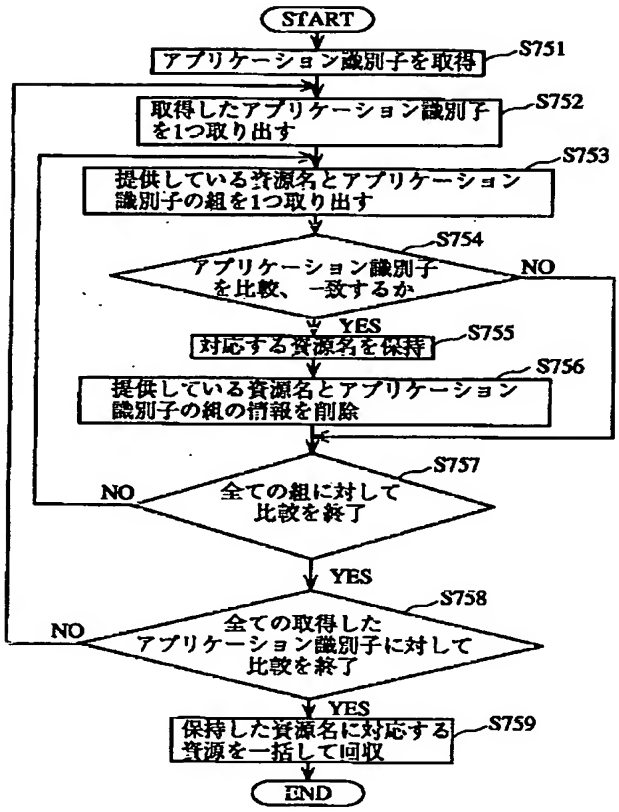
【図4】



【図6】



【図7】



【図9】

```
public interface resourceCollectionListener{
  public void update();
}
```

【図10】

```
public class applicationProxy {
  public string name;  // application name

  public int getStatus() {
    // get the status of application,pause or destroy
  }

  public addListener(resourceCollectionListener 1){
    // register resource collection instance
  }

  public removeListener(resourceCollectionListener 1){
    // remove resource collection intance
  }
}
```

【図11】

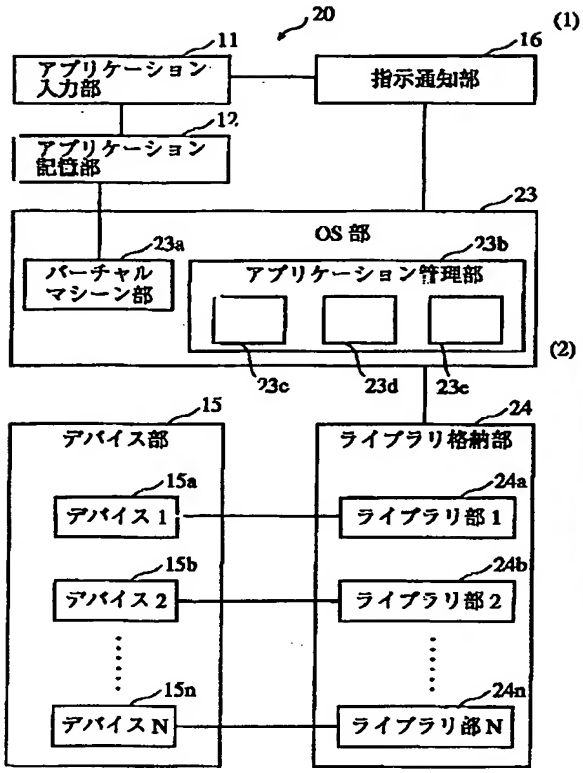
```
public class fileCollectionListener
        implements resourceCollectionListener{
  String name;  // filename
  ApplicationProxy app;
                // Application information instance

  public void update() {
    // release file which has filename
  }
}
```

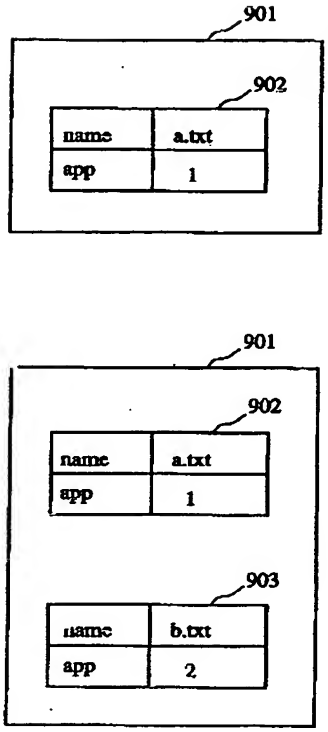
【図25】

アプリケーションID	クラスローダー
1	classloader1
2	classloader2

【図8】



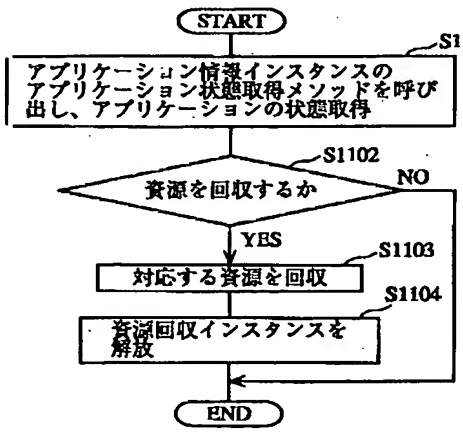
【図12】



【図27】

アプリケーションID	ファイル
1	A.txt B.txt
2	C.txt

【図14】



【図16】

(1)

アプリケーションID	タスクID	Thread ID
1	201	1,2
2	202	4,5,6

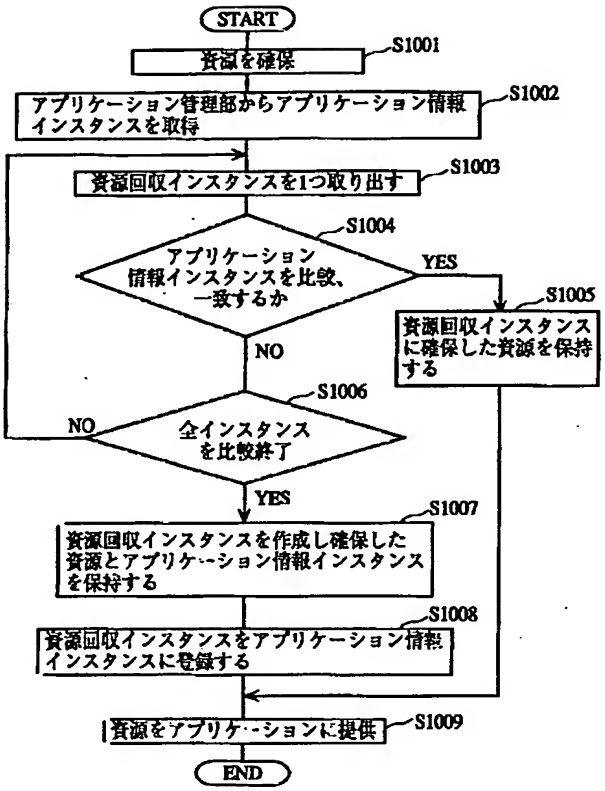
(2)

アプリケーションID	タスクID	Thread ID
1	201	1,2,7
2	202	4,5,6

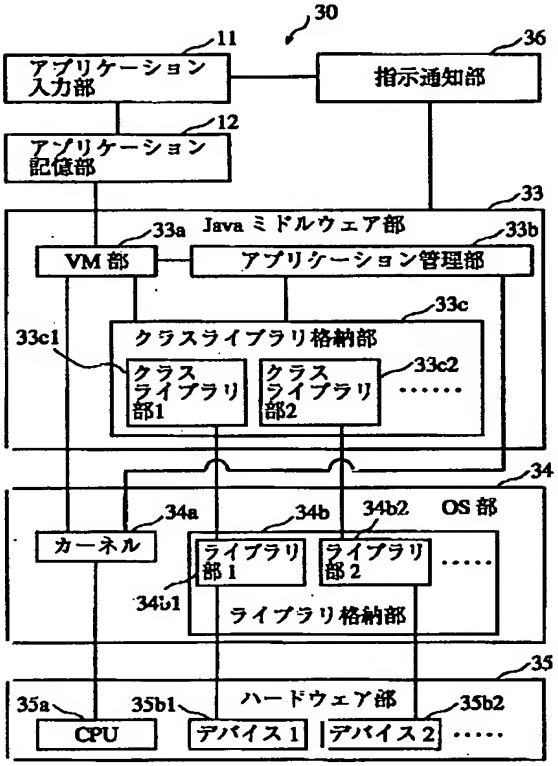
(3)

アプリケーションID	タスクID	Thread ID
1	201	1,2,7
2	202	4,6

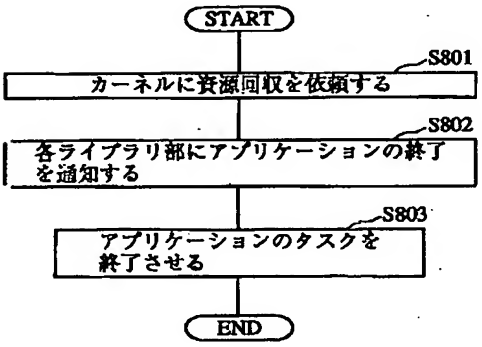
【図１３】



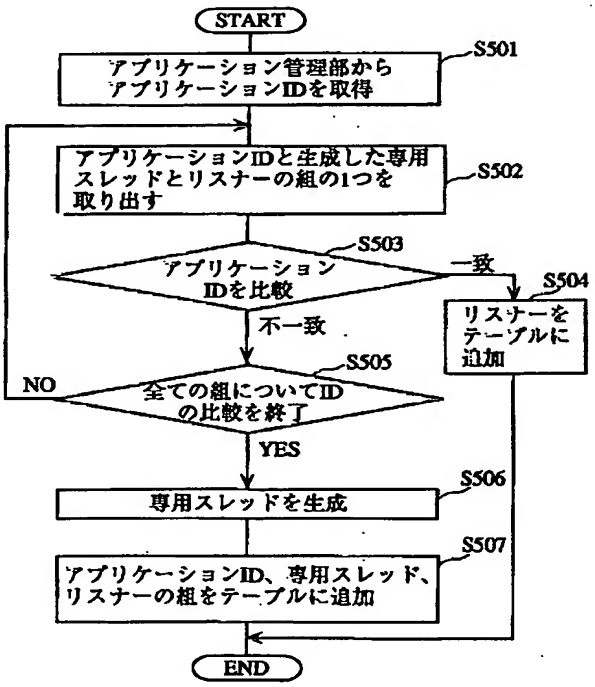
【図１５】



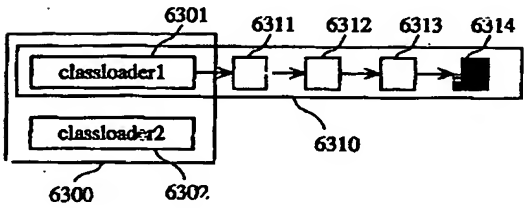
【図１７】



【図１８】



【図２６】



【図19】

(1)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4

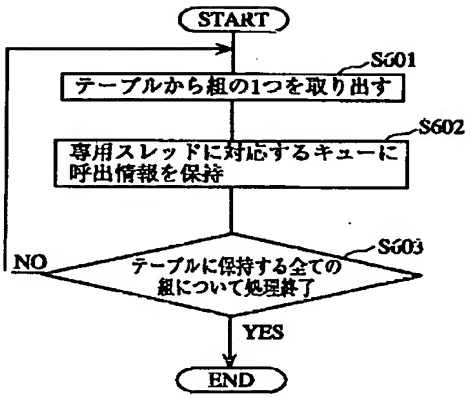
(2)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4
2	thread20	L6

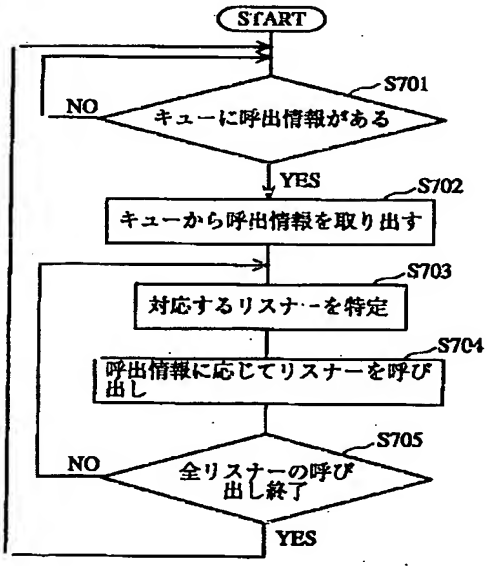
(3)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
2	thread20	L6

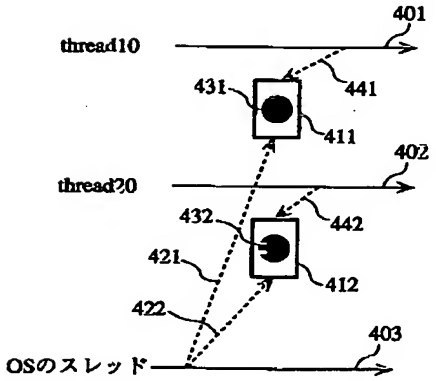
【図20】



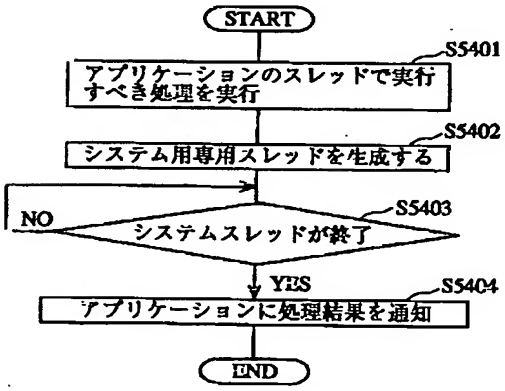
【図21】



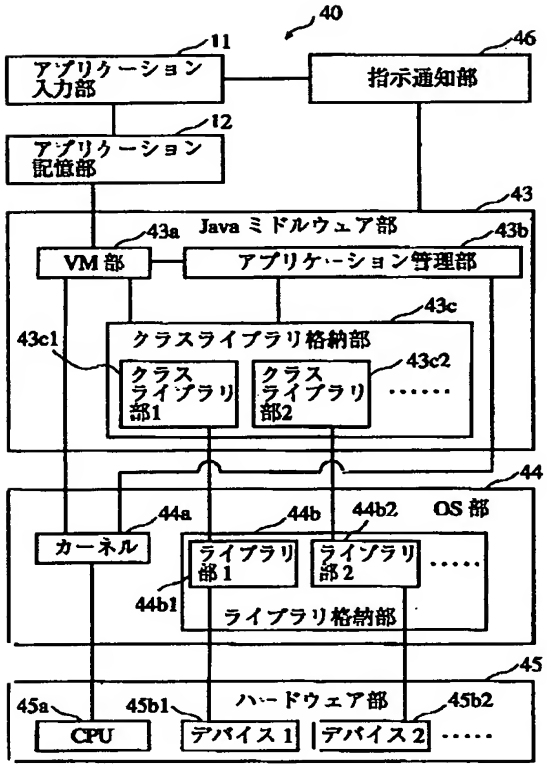
【図22】



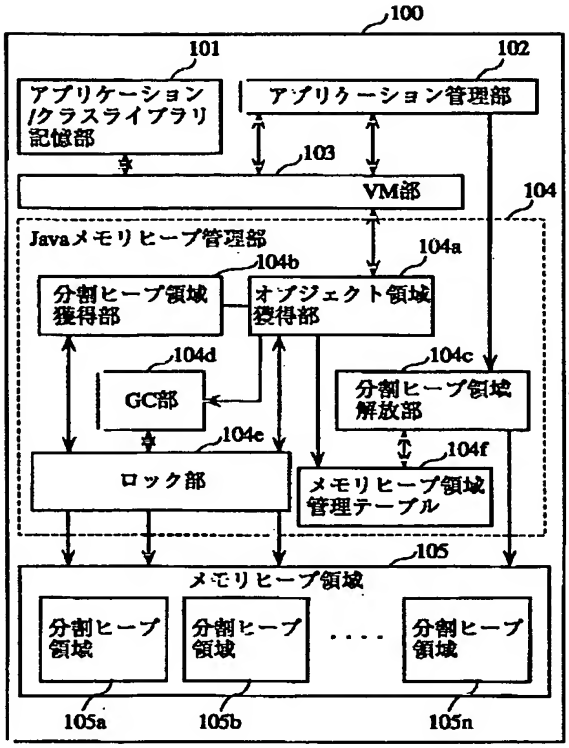
【図23】



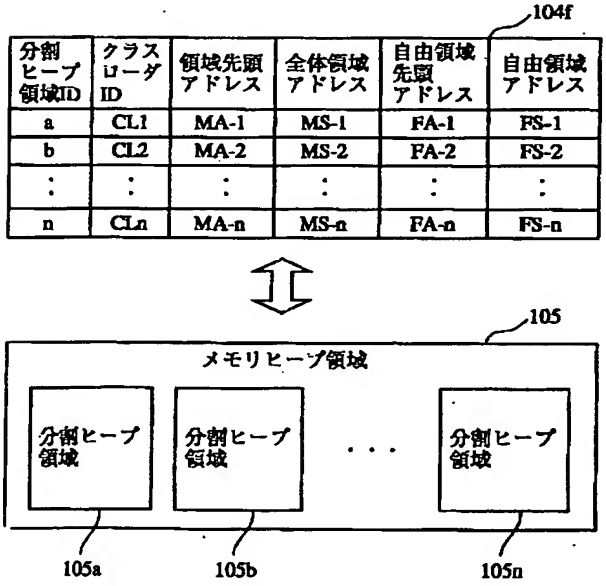
【図24】



【図28】

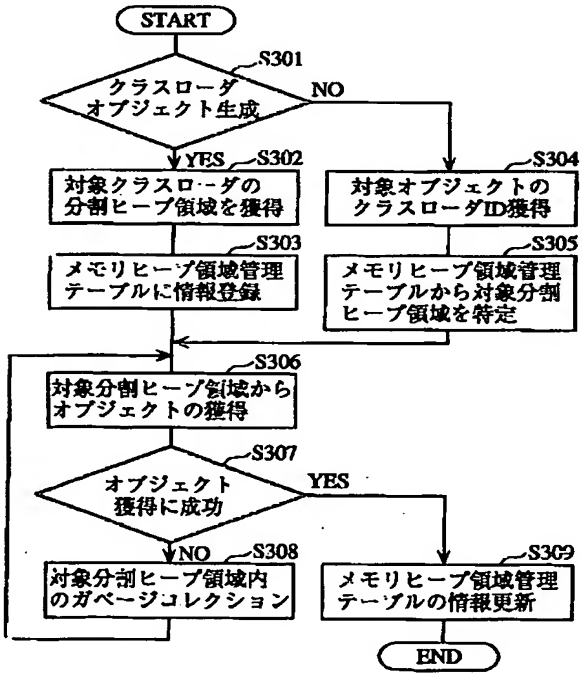


【図29】

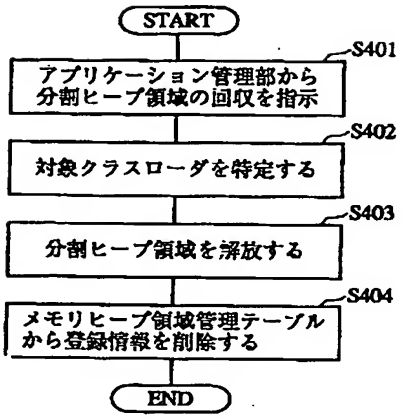


分割ヒープ領域ID	クラスローダID	領域先頭アドレス	全体領域アドレス	自由領域先頭アドレス	自由領域アドレス
a	CL1	MA-1	MS-1	FA-1	FS-1
b	CL2	MA-2	MS-2	FA-2	FS-2
:	:	:	:	:	:
n	CLn	MA-n	MS-n	FA-n	FS-n

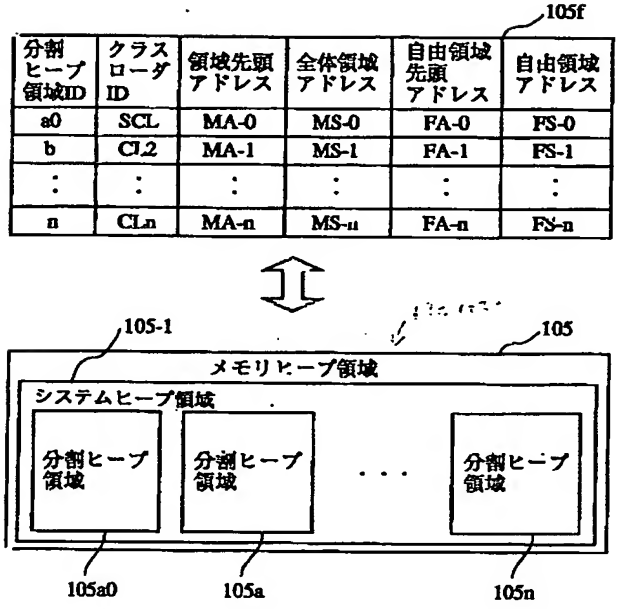
【図30】



【図31】



【図32】



フロントページの続き

(72)発明者　平本　建志
大阪府門真市大字門真1006番地　松下電器産業　株式会社内
(72)発明者　久保岡　祐子
大阪府門真市大字門真1006番地　松下電器産業　株式会社内
(72)発明者　土井　繁則
大阪府門真市大字門真1006番地　松下電器産業　株式会社内

Fターム(参考)　5B060 AA10
5B098 GD03 GD07 GD14 GD22